



UNIVERSIDAD DE LA RIOJA

TESIS DOCTORAL

Título
Democratizing Deep Learning methods by means of AutoML tools
Autor/es
Manuel García Domínguez
Director/es
César Domínguez Pérez y Jonathan Heras Vicente
Facultad
Facultad de Ciencia y Tecnología
Titulación
Departamento
Matemáticas y Computación
Curso Académico

Tesis presentada como compendio de publicaciones. La edición en abierto de la misma NO incluye las partes afectadas por cesión de derechos



Democratizing Deep Learning methods by means of AutoML tools, tesis doctoral de Manuel García Domínguez , dirigida por César Domínguez Pérez y Jonathan Heras Vicente (publicada por la Universidad de La Rioja), se difunde bajo una Licencia Creative Commons Reconocimiento-NoComercial-SinObraDerivada 3.0 Unported. Permisos que vayan más allá de lo cubierto por esta licencia pueden solicitarse a los titulares del copyright.

Democratizing Deep Learning methods by means of AutoML tools

Manuel García Domínguez

Supervisors: Dr. D. César Domínguez Pérez
Dr. D. Jónathan Heras Vicente

A thesis submitted in fulfillment of the requirements
for the degree of Doctor of Philosophy

Universidad de La Rioja
Departamento de Matemáticas y Computación



**UNIVERSIDAD
DE LA RIOJA**

Logroño, September 2022

This work was partially supported by Ministerio de Economía, Industria y Competitividad [MTM2017-88804-P], Ministerio de Ciencia e Innovación [PID2020-115225RB-I00 / AEI / 10.13039/501100011033], Agencia de Desarrollo Económico de La Rioja [ADER-2017-I-IDD-00018] and a FPI grant from Community of La Rioja 2018.

Agradecimientos

Este trabajo está dedicado a todas las personas que me han acompañado a lo largo de estos años.

Para comenzar, quiero dar las gracias a Jónathan Heras y César Domínguez, quienes me han acompañado y orientado durante esta etapa. Gracias por vuestra comprensión y empatía, por tener siempre vuestra puerta abierta y ayudarme a ver el camino cuando era difuso.

También quiero agradecer a mi grupo de investigación, que me recibieron con los brazos abiertos y han sido un gran apoyo.

No me olvido de mis compañeros y compañeras de doctorado, que sin duda han sabido entenderme durante todo este tiempo y que se han convertido en grandes amigos y amigas.

Agradezco también toda la ayuda recibida por el departamento de Matemáticas y Computación, que siempre me ha facilitado las herramientas necesarias para hacer mi trabajo lo más cómodo posible.

Por último, y no por ello menos importante, doy las gracias a mi familia, amigos y amigas; que aun sin saber entender en qué consistía mi trabajo, siempre han sabido como ayudarme. Especialmente, a mi madre, a Andrea y a mis gatos.

A todas y todos, gracias.

Resumen

Las técnicas de Inteligencia Artificial, y en concreto de aprendizaje profundo, o en inglés *Deep Learning*, se han convertido en el estado del arte para lidiar con problemas de Visión por Computador en casi cualquier ámbito. Este crecimiento se debe a la gran cantidad de imágenes que se capturan diariamente, el incremento de la capacidad de cálculo gracias al desarrollo de hardware específico, y el acceso libre a las herramientas necesarias para crear modelos de *Deep Learning*. A pesar de su éxito, los métodos de *Deep Learning* presentan una serie de problemas. Esta tecnología no es sencilla de utilizar y requiere de cierta experiencia y conocimientos técnicos. Además, no hay siempre un algoritmo, o una librería, que consiga los mejores resultados para cualquier situación, por lo que es necesario conocer y probar múltiples de ellos. Otro problema añadido es la necesidad de obtener un gran número de imágenes anotadas para poder construir modelos precisos. Esto puede ser un reto en contextos como el biomédico donde puede ser difícil conseguir un número suficiente de imágenes, y donde el proceso de anotación es costoso y requiere de conocimiento experto. Por último, una vez que un modelo de *Deep Learning* se ha construido, este debe ser capaz de generalizar a contextos no previstos en el momento de su construcción; sin embargo, en muchas ocasiones los modelos no funcionan correctamente cuando se usan imágenes de un dominio, o estilo, diferente al que se usó originalmente para entrenar dichos modelos. Este problema se conoce como el problema del cambio de dominio.

El objetivo de nuestro trabajo ha sido dar solución a los problemas mencionados anteriormente mediante el desarrollo de técnicas y herramientas que sean accesibles a la mayor cantidad de usuarios posibles. En primer lugar hemos desarrollado herramientas que permiten crear de manera sencilla modelos precisos de *Deep Learning* para la clasificación y la detección de objetos en imágenes. Para ello se han empleado técnicas de AutoML que buscan de forma automática el mejor modelo para un conjunto de imágenes dadas. Además, hemos desarrollado un método para aplicar aumento de datos a distintos problemas de Visión por Computador. Este método ha sido implementado en una herramienta que permite generar datasets de imágenes lo suficientemente grandes para alimentar a los modelos de *Deep Learning* en distintas tareas de Visión por Computador como son la clasificación, la detección de objetos o la segmentación semántica de imágenes y vídeos. Además de herramientas para facilitar la creación de modelos de *Deep Learning*, se ha desarrollado una herramienta que mediante técnicas de traducción desparejada de imágenes (en inglés, *unpaired image-to-image translation*) y transferencia de estilos (en inglés, *style transfer*), permite lidiar con el problema del cambio de dominio en cualquier problema de Visión por Computador. Es importante notar que

no solo hemos desarrollado métodos y herramientas desde un punto de vista teórico, sino que todo el conocimiento adquirido durante el desarrollo de dichas herramientas ha servido para abordar problemas biomédicos reales como son la segmentación de esferoides, la clasificación y segmentación de imágenes de motilidad, o la predicción de enfermedades de la retina a partir de imágenes del fondo del ojo. Finalmente, los conocimientos obtenidos al resolver estos problemas biomédicos nos han servido para mejorar las herramientas desarrolladas.

Abstract

Artificial Intelligence, and specifically Deep Learning, methods have become the state-of-the-art approach to deal with Computer Vision problems in almost any field. This growth is due to the large amount of images that is produced in a daily basis, the increment of calculation capacity thanks to the development of specific hardware, and the open-source nature of the tools that allow us to build Deep Learning models. Despite their success, Deep Learning methods have several drawbacks. This technology might be difficult to use and requires some experience and technical knowledge. In addition, there is not always an algorithm, or library, that produces the best results for all the situations; hence, it is necessary to know and try different alternatives. Moreover, Deep Learning methods require a large number of labeled images to produce accurate models. This might be a challenge in contexts like biomedicine where it is difficult to acquire large enough datasets of images, and the annotation of those images require expert knowledge. Finally, once Deep Learning models are built, they should be able to generalize to contexts that were unforeseen during their construction. However, in many cases, models do not work properly when they are used with images from a domain, or style, different to the one used for training those models — this is known as the domain shift problem.

The goal of our work has been to tackle the aforementioned problems by means of techniques and tools that are user-friendly. First of all, we have developed tools that allow users to create accurate Deep Learning models for image classification and object detection tasks. To this aim, we have applied AutoML techniques that automatically search the best model for a given dataset of images. Moreover, we have developed a method to apply data augmentation to several Computer Vision problems. Such a method has been implemented in a tool that allows users to generate large enough datasets of images to feed Deep Learning models in several Computer Vision tasks such as image and video classification, object detection or semantic segmentation. In addition to tools that simplify the construction of Deep Learning models, we have developed a tool that tackles the domain shift problem by means of unpaired image-to-image translation methods and style transfer techniques. It is worth noting that we have not only developed methods and tools from a theoretical point of view, but all the knowledge acquired during the development of those tools has been applied to deal with actual biomedical problems such as spheroid segmentation, the classification and segmentation of motility images, or the diagnosis of retinal diseases from fundus images. Finally, the experience provided by tackling actual problems has served to improve the developed tools.

Contents

Publications	1
1 Introduction	3
1.1 Motivation	3
1.2 Objectives	11
2 Results and discussion	13
2.1 Objective 1: Facilitating the construction of Deep Learning models . .	13
2.2 Objective 2: A General Image Augmentation Library	23
2.3 Objective 3: Dealing with the Domain Shift Problem	29
2.4 Objective 4: Applications to biomedical problems	36
3 Contributions	45
3.1 FrImCla: A Framework for Image Classification Using Traditional and Transfer Learning Techniques	46
3.2 UFOD: An AutoML Framework for the Construction, Comparison, and Combination of Object Detection Models	47
3.3 CLoDSA: a tool for augmentation in classification, localization, detection, semantic segmentation and instance segmentation tasks	48
3.4 Neural Style Transfer and Unpaired Image-to-Image Translation to deal with the Domain Shift Problem on Spheroid Segmentation	49
3.5 Prediction of Epiretinal Membrane from Retinal Fundus Images Using Deep Learning	50
3.6 MotilityJ: An open-source tool for the classification and segmentation of bacteria on motility images	51
4 Conclusions and further work	53
5 Conclusiones y trabajo futuro	57

Publications

This memoir is made up of a compendium of scientific papers. These papers are attached as appendixes of this document and are listed as follows.

- A. Casado-García, C. Domínguez, M. García-Domínguez, J. Heras, A. Inés, E. Mata, and V. Pascual. “CLODSA: A Tool for Augmentation in Classification, Localization, Detection, Semantic Segmentation and Instance Segmentation Tasks”. *BMC Bioinformatics* 20(323). 2019. DOI: 10.1186/s12859-019-2931-1.
- M. García-Domínguez, C. Domínguez, J. Heras, E. Mata, and V. Pascual. “FrIm-Cla: A Framework for Image Classification using Traditional and Transfer Learning Techniques”. *IEEE Access*. 2020. DOI: 10.1109/ACCESS.2020.2980798.
- M. García-Domínguez, C. Domínguez, J. Heras, E. Mata, and V. Pascual. “UFOD: An AutoML Framework for the Construction, Comparison, and Combination of Object Detection Models”. *Pattern Recognition Letters* 145, pp. 135-140, 2021, DOI: 10.1016/j.patrec.2021.01.022

The rest of this memoir is organized as follows. In the first chapter, we present the motivation of our work, the problems that have been addressed, and the objectives that we would like to achieve with this work. In the second chapter, we show and discuss the results that we obtained with our work, and in Chapter 3 we summarize our main contributions. To conclude, we present the conclusions of this work and future lines of research. The code associated with this memoir can be found in the following repository: <https://github.com/ManuGar/Thesis>.

Chapter 1

Introduction

In this chapter, we provide the necessary background to understand the motivation of our work. Namely, we introduce what is Deep Learning, present some drawbacks of Deep Learning methods for Computer Vision tasks, and review the works in the literature that have addressed those limitations. Finally, we establish the goals that have driven our research.

1.1 Motivation

In the last decade, we have seen a revolution in several fields thanks to the application of Artificial Intelligence, and, more concretely thanks to, *Deep Learning* methods [1]. Artificial Intelligence is a field of Computer Science that is focused on designing intelligent systems [2]. The main goal of Artificial Intelligence is the creation of systems that are autonomous and adaptable. Among Artificial Intelligence techniques, we can highlight a subset of them called *Machine Learning* methods that, instead of being explicitly programmed, learn from data [3]. The impact of Machine Learning methods has considerably grown thanks to the huge amount of data that is produced in a daily basis, the increase of the calculation capacity, and the fact that the community makes, in general, its algorithms and models freely available for use [1]. There are several kinds of Machine Learning techniques such as reinforcement, unsupervised or semi-supervised learning methods; however, the most well-known procedures in this field are supervised learning methods. In supervised learning methods, there are two main ingredients: a labeled dataset (that consists of pairs of input and expected output), and an algorithm (for instance, a decision tree or a neural network) that has a set of configurable parameters, also known as weights. During the learning process, elements of the labeled dataset are shown to the algorithm and its parameters are modified to produce the expected output thanks to an optimization algorithm [4]. The result of the learning process is a model; that is, an algorithm with a fixed set of weights, that can be used for inference. Among supervised learning algorithms, we can highlight *neural networks* since they are the basis for Deep Learning models. Neural networks are based on a collection of artificial neurons that are connected. Each connection can

transmit a signal to other neurons. An artificial neuron receives signals then processes them and can signal neurons connected to it. Deep Learning is a subset of Machine Learning where the “*deep*” comes from the number of layers that form those neural networks and are responsible for extracting the information from the data to learn a useful representation [5]. The main difference between Machine Learning and Deep Learning methods is that in Machine Learning methods the representation of the data is fixed by the developer; and, on the contrary, Deep Learning methods learn the data representation that is useful for a given task without human intervention. Among the fields that have taken advantage of the advances produced thanks to Deep Learning, we can highlight Computer Vision.

Computer Vision is a field of Artificial Intelligence that is focused on extracting and analyzing information from images and videos to solve real life problems [6]. Roughly speaking, the objective of Computer Vision techniques is to teach computers to “see”. Deep Learning methods have been successfully used in Computer Vision tasks because these methods learn a hierarchical representation of images, similar to the human, to solve tasks such as image classification within X-ray baggage security [7] or in the classification of gastrointestinal diseases [8].

Among Computer Vision tasks, we can highlight three tasks that have lots of applications: image classification, object detection and semantic segmentation (see Figure 1.1). Image classification consists of determining the class to which an image belongs. Object detection is a more complex task since it is focused on locating the position of the objects in an image and determining their classes. The last task, semantic segmentation, is the most complex. In semantic segmentation, each pixel of the image is classified using a fixed set of categories. Currently, the state-of-the-art approach to tackle these tasks are Deep Learning models, and namely Convolutional Neural Networks (CNNs) [1].



Figure 1.1: Common Computer Vision tasks. The images are available under a Creative Commons License

In spite of its success, the application of Deep Learning methods for image classification, object detection and semantic segmentation is not straightforward [1]. Deep learning methods need some experience and knowledge of the underlying technology because it is complex and difficult to use. Moreover, there are several Deep Learning libraries and algorithms that can be used to solve the same problem, and there is not a single algorithm that always gets the best results [3]. In addition, Deep Learning algorithms also need huge datasets to get good results [9] and, they fail to generalize when

applied to out-of-domain data distributions [10, 11]. In the next sections, we provide a detailed description of these problems and the solutions that have been proposed in the literature.

1.1.1 Democratization of Deep Learning tools

The construction of Deep Learning models requires some knowledge about their training methods, and the tools that implement them [1]. In addition, there are different libraries and algorithms that address the same problem (for instance, CNNs such as ResNet [12] or Efficientnet [13] can be used for tackling images classification tasks, or YOLO [14] and FasterRCNN [15] can be used for object detection problems). Moreover, Deep Learning algorithms have a set of parameters known as hyperparameters that are not learned, but must be fixed by the user. So, it is convenient to try several configurations and alternatives to find the best approach for each problem [16]. As there are many different alternatives, and users from different disciplines normally are not experts in these technologies, AutoML systems have been developed to democratize the use of Machine Learning methods [17]. AutoML tools try to simplify the construction and usage of Machine Learning models for domain experts that have a limited Machine Learning background [18].

There are two different types of AutoML tools: AutoML tools for structured data, and AutoML for non-structured data (for instance, images, text or videos). AutoML tools for structured data are focused on hyperparameter tuning, trying to find the best combination of hyperparameters and classical Machine Learning algorithms. In the literature, we can find several AutoML tools for structured data like SMAC [19], Auto-Weka [20] or Auto-sklearn [21]. These systems select the algorithm which produces the best results after performing several tests on the problem that is studied. AutoML tools for structured data have the drawback that they cannot be used directly with images or texts; therefore, other kinds of tools must be used.

AutoML techniques for non-structured data are mainly focused on automatically designing neural deep architectures [22]. This is known as Neural Architecture Search (NAS), and it is used to find new Deep Learning architectures optimized for a given problem. NAS methods have improved the results on problems such as image classification or object detection [23], and there are several tools that can apply those methods; for instance, Auto-Keras [24], AutoML cloud services [25] or Google AutoML [26]. Most of these tools are mainly oriented towards image classification tasks, which limits their use. In addition, these tools require a high number of images to train the models and take too much time; for instance, 500 Graphics Processing Unit, or GPUs, were used during 4 days to train a model for the CIFAR-10 dataset [27]. Another aspect to take into account in the case of AutoML cloud services is the privacy of our projects since we have to upload the images to servers that are not under our control. Therefore, in order to facilitate the construction of Deep Learning models for Computer Vision tasks, it is necessary the development of tools that can be applied to different problems, are easy to use, and produce effective models in a reasonable amount of time and using a limited amount of resources. In addition, we have to provide these AutoML tools a sufficient amount of images.

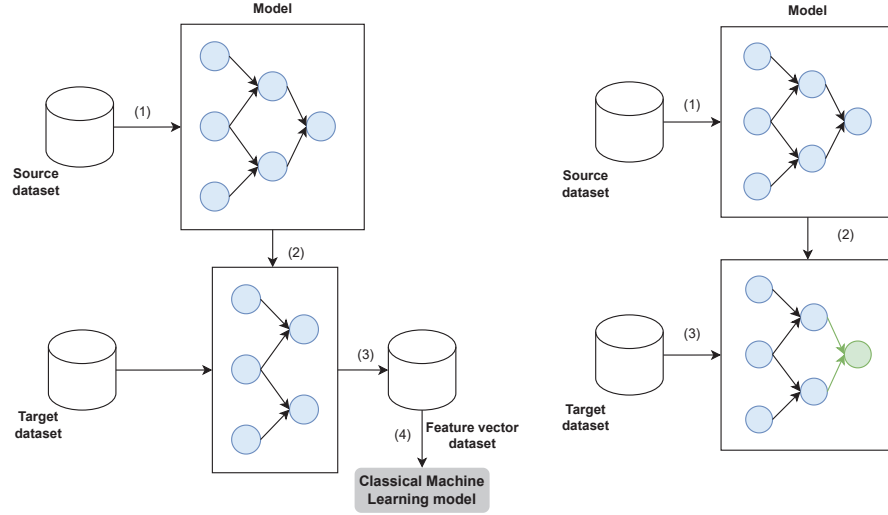


Figure 1.2: Transfer learning approaches for image classification. **LEFT.** (1) The source dataset is used to train the model. (2) The trained model is used but removing the last layer. (3) We use the model to generate a feature vector dataset from the target dataset. (4) A classic machine learning model is trained with the feature vector dataset. **RIGHT.** (1) The source dataset is used to train the model. (2) We use the model by replacing the last layer. (3) The target dataset is used to retrain the model.

1.1.2 The amount of available images

Deep Learning algorithms need millions of images to be trained from scratch [28]. In some cases, getting such an amount of images can be really difficult because of a tight budget to obtain more samples, the need to perform invasive medical procedures, or destructive image processes that limit the images that can be acquired. An additional problem arises since these images need to be annotated; a manual process that might require an expert, and takes a considerable amount of time. For example, in the COCO dataset, that includes 80K images manually annotated for semantic segmentation, it took 22 worker hours per 1,000 segmentations [29].

One approach to solve the lack of enough images is *transfer learning* [30]. This technique uses a model trained in a source task in a new target task [31–34]. There are two different approaches to use transfer learning for image classification: either as a feature extractor or by using fine-tuning. Using transfer learning as feature extractor for image classification consists in using a network pre-trained on a big source dataset, like ImageNet [35], and using the output of the last but one layer of such a pretrained network as input of classical Machine Learning algorithm. Hence, each image of the target dataset is fed to the source network and a feature vector is extracted, and then it is possible to use all classical Machine Learning algorithms. The fine-tuning approach also starts from a pre-trained network but the model architecture is modified by changing the last layer, and providing a new layer adapted to our particular task.

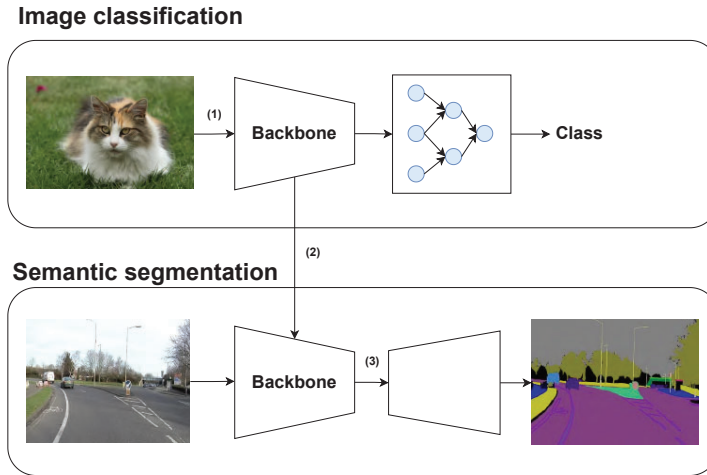


Figure 1.3: Transfer learning on semantic segmentation tasks. (1) The network is trained on an image classification task. (2) The backbone of this network is used in a new target dataset as part of the new network. (3) The new network with the backbone is trained on a semantic segmentation problem.

With this “new” network, the model is retrained with the target dataset to obtain the new model, see Figure 1.2. In both transfer learning approaches, there are not any tool that facilitates their use by non-expert users since all of them require some knowledge about specialized libraries like Keras [36], PyTorch [37] or fastai [38]. Moreover, there are several architectures that can be used as source models, and it might be worth exploring several of them. In order to use transfer learning in other problems, like object detection or segmentation, we can use a model pre-trained for image classification as the backbone of the network that will be employed to extract relevant features for the given problem, see Figure 1.3. Namely, a backbone that has been trained on the source dataset will be used in a new dataset to make it easier to abstract new patterns. In this way, the new network, by using the backbone trained on a previous task, is able to obtain patterns faster and improve the training results.

Another way to solve the problem of the lack of images for Deep Learning methods is *data augmentation* [9, 39]. This technique is based on increasing the number of available images through transformations such as rotations or color changes. It is worth noting that some transformations do not make sense in some contexts; for instance if we modify the color of a cat we do not modify the content of the image, but if we make the same color change in a medical image, we could modify the interpretation of such an image. In addition, for Computer Vision tasks such as object detection or semantic segmentation, the annotation of the image also changes when some transformations are applied, see Figure 1.4. Over the years, several tools have been developed for data augmentation. Initially they were basic tools that could only be used for image classification. Moreover, some of these tools did not work with all annotation formats or did not work with videos. But they have been upgraded over the years and most libraries

have solved those problems. Some of those libraries are Imgaug [40], Augmentor [41], Albumentations [42] and autoaugment [43].

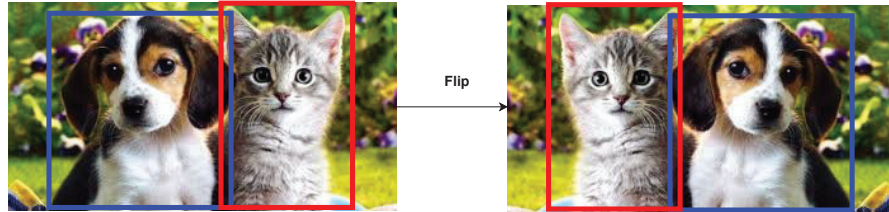


Figure 1.4: Data augmentation example for object detection. The images are available under a Creative Commons License.

Most transformations provided by data augmentation libraries are either geometric or color transformations. A different kind of data augmentation method is based on Generative Adversarial Networks (GANs) [44]. This technology is a bit peculiar since it is based on the fact we build a model that it is formed by two neural networks that “compete” with each other (see Figure 1.5). The first network, called the *generator*, is focused on the creation of realistic images. The second network, called the *discriminator*, is focused on deciding whether an image is real or was created by the generator. In the training process, we see a “competition”, where the generator improves the images it creates to deceive the discriminator, that in turn is improved to detect fake images. GANs can be used to change the style of images [45] and generate new images that can be used in combination with little datasets for data augmentation [46]. The main disadvantage of GANs is that they need huge datasets to be trained to generate images with good quality. Moreover GANs like other Deep Learning techniques are difficult to train and need a lot of execution time to train them [47].

In order to facilitate the training process of GANs for generating new images, an approach called *image-to-image* translation has emerged [48]. Image-to-image translation methods take images from one domain and transform them so they have the style of a different domain, see Figure 1.6. This approach has the drawback that needs paired data from source and target domains, but this issue has been fixed with *unpaired image-to-image* translation methods [50], see Figure 1.7. This new technology does not require paired images and can translate images from domain *A* to domain *B*, and vice versa. This method uses a dataset from domain *A* to learn the style of the images and then uses a dataset from domain *B* to translate the images of such a dataset into the style of domain *A* and vice versa. Among unpaired image-to-image translation methods we can highlight techniques like CycleGAN [49], DualGAN [51], ForkGAN [52], GANILLA [53], CUT [50], and FASTCUT [50] that have been used successfully as image augmentation methods [54]. The main limitation to use those techniques is the lack of a library that facilitates the use of those techniques for non-expert users.

Up to now, we have presented the limitations for building Deep Learning models, but

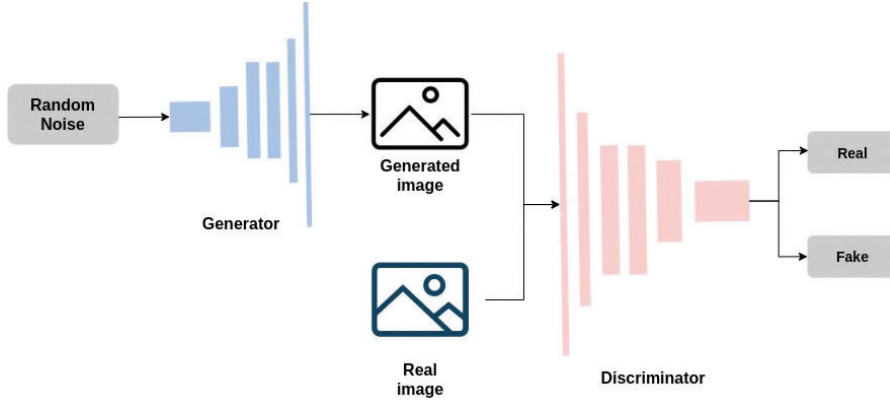


Figure 1.5: GAN training process. Generator creates realistic images, whereas the discriminator determines whether the image is real or generated by the generator.

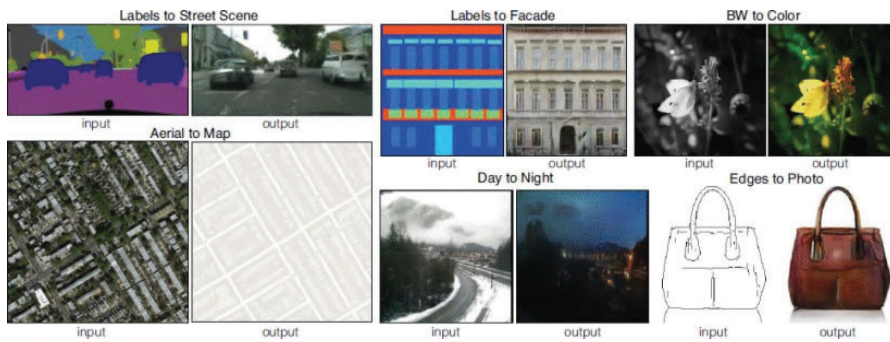


Figure 1.6: Several image-to-image translation problems [48].

once we have trained them, we should check whether they are capable of generalizing; that is, check whether the models can deal correctly with images that were not seen during the training process.

1.1.3 Generalization of models

The last challenge of Deep Learning models that we are going to face in this work is known as *domain shift* (also known as *distribution shift*) [10, 11]. This problem arises when the data distribution of the dataset used for training is not the same to the data that the model encounters when it is deployed, see Figure 1.8. This is a common problem when working with domains that have a lot of variability, such as autonomous driving, where we have different climatic conditions [55]; or in biomedical domains, where there is a great variability in experimental conditions, the equipment or settings (for instance, the microscope used, or the focus and colors used to obtain the images) [56]. This generalization problem can be mitigated by combining several

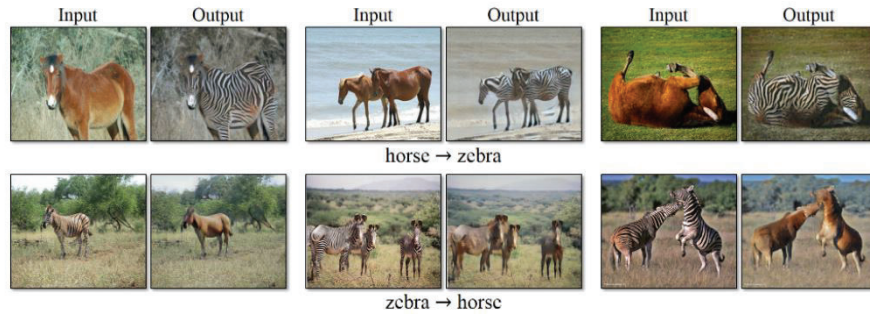


Figure 1.7: Image style change example of using CycleGAN. Object transfiguration between horses and zebras [49].

datasets from multiple sources [57] or by using data augmentation, but it is impossible to include every distribution in the training dataset.

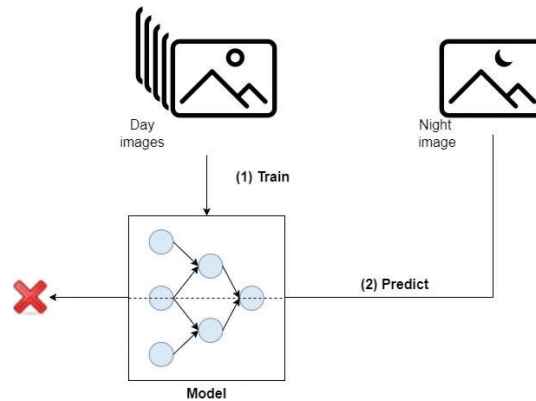


Figure 1.8: Generalization problem with images with a completely different domain or style

Another solution to the domain shift problem consists in applying transfer learning; namely, by retraining a model with data from the new distribution, but this requires an annotated dataset of the new distribution, and the resources and knowledge to train the new model. A novel way to solve the domain shift problem is the application of unpaired image-to-image translation methods [58], this is illustrated in Figure 1.9.

As we have previously mentioned, unpaired image-to-image translation methods translate an image from a domain A to a domain B , and vice versa, in the absence of paired examples. This approach has been already employed in several medical segmentation tasks; for instance, CycleGAN was used to change the style of a test set of magnetic resonance images for segmenting the left ventricle [59], it was also used to modify the style of x-ray images in the segmentation of radiographs [60], and to generate the medical images that are used in the development of a project for the segmentation

of magnetic resonance imaging (MRI), abdominal CT and MRI, and mammography X-rays [61]. All these works were based on variants of CycleGAN [49]. This approach to deal with domain shift poses two challenges. First, both datasets (the dataset used for training, and the dataset of the new domain) must be available, and this might be an issue due to privacy concerns [62]; and, secondly, CycleGAN variants must be trained, a process that demands the usage of GPUs and might be challenging for several users due to the difficulties of training GAN models [47], therefore, other methods should be studied.

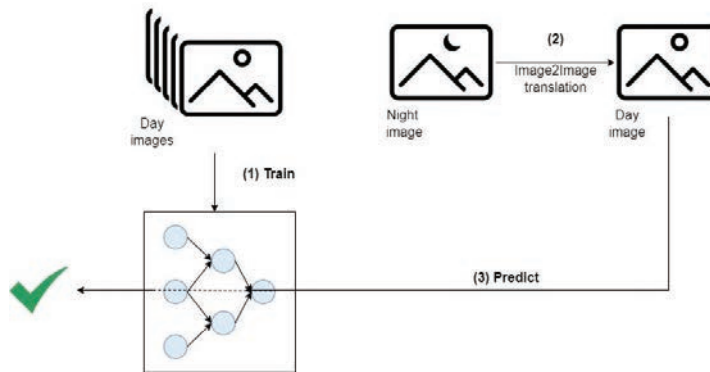


Figure 1.9: Image to image translation workflow to deal with the domain problem shift.

1.2 Objectives

In the previous section, we have presented some limitations that are faced by Deep Learning methods when applied to Computer Vision tasks. The aforementioned limitations make it difficult for non-expert users to take advantage of Deep Learning methods. Therefore, the main goal of our work is the democratization of Deep Learning techniques by developing new methods and tools that help non-expert users in the construction and usage of Deep Learning models for Computer Vision. To reach such a goal, we have fixed the following objectives:

- O.1. Develop AutoML tools that facilitate the construction and use of Deep Learning models with limited resources for image classification and object detection, and that can be generalized to other Computer Vision tasks like semantic segmentation.
- O.2. Develop a tool that allows anyone to generate the number of images required to train Deep Learning models in a wide variety of Computer Vision tasks.
- O.3. Develop techniques and tools to address the domain shift problem in the context of Computer Vision tasks.
- O.4. Test the suitability of the tools and methods developed in the previous objectives with real life problems.

Chapter 2

Results and discussion

In this chapter, we present how we have achieved the objectives introduced in the previous section. A summary of our contributions is provided in Table 2.1.

Table 2.1: Results of this work

Nb	Results	Objective	Publications
R1	Creation of a user-friendly tool for training and deploying image classification models.	O.1	[63–65]
R2	Creation of a user-friendly tool for training and deploying of object detection models.	O.1	[66, 67]
R3	Creation of a tool that helps in the task of data augmentation on computer vision problems that are data demanding but have small image datasets.	O.2	[68]
R4	Creation of a tool that allows anyone to modify the style of a dataset to the style of the dataset that the model was trained.	O.3	[69]
R5	Development of models for predicting epiretinal membrane using Deep Learning.	O.4	[70]
R6	Development of a tool for the classification and segmentation of bacteria on motility images.	O.4	[71]

2.1 Objective 1: Facilitating the construction of Deep Learning models

The first objective of this work was the simplification of the process to construct Deep Learning models for image classification and object detection tasks. This has been achieved thanks to the development of two AutoML tools called: FrImCla and UFOD.

2.1.1 FrImCla: An AutoML tool for image classification

Currently, AutoML tools for building Deep Learning models for image classification are based on the application of NAS methods. As we have previously mentioned, this approach has several drawbacks since it is data demanding, time consuming and hard

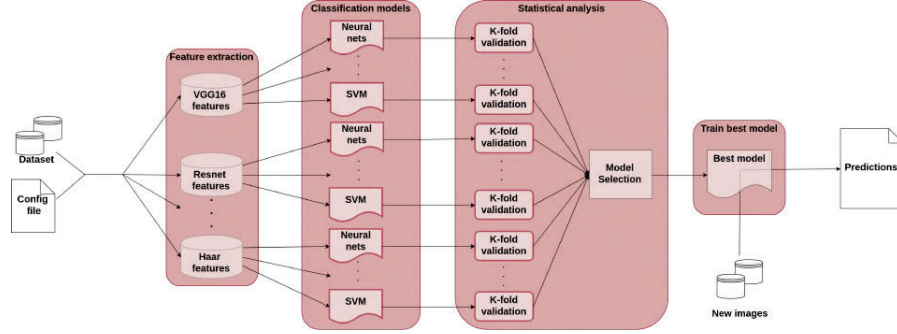


Figure 2.1: Workflow of the FrImCla framework

to use for non-expert users [22]. On the contrary, we have approached this problem inspired by tools like SMAC [19], Auto-Weka [20] or Auto-sklearn [21]. In particular, we have built a tool, called FrImCla, that automatizes the construction of models for image classification by using both Deep Learning models as feature extractors, and also traditional Computer Vision feature extractors.

FrImCla (that stands for Framework for Image Classification) is an open-source library that has been implemented in Python. This framework relies on several third-party open-source libraries that have been employed and tested by large communities; namely, scikit-learn [72], for machine learning methods and parallelization techniques; scikit-multilearn [73], for multi-label classification algorithms; OpenCV [74], to extract traditional computer vision features; Keras [36], to extract Deep Learning features; and cPickle [75], to serialize objects.

The workflow of the FrImCla framework is depicted in Figure 2.1 and can be summarized as follows. First of all, the user selects the image dataset to be studied and some configuration parameters (mainly the feature extraction and Machine Learning methods to be used). Subsequently, FrImCla extracts features from the dataset using the set of fixed feature extractor methods given by the user; and a dataset of features is created for each of them. On these datasets of features, FrImCla trains the selected Machine Learning algorithms, and a statistical analysis is conducted to select the best combination of features and Machine Learning algorithm. From such a winner combination, a model is created for further usage. Apart from the first step — that is, the selection of feature extraction methods and algorithms to be tried — the rest of the process is carried out automatically by FrImCla without any user intervention. In the rest of this section, we provide a brief overview of FrImCla’s features.

FrImCla description

FrImCla has been developed for different types of users. Non-expert users have at their disposal tools that make its execution easier (in particular we have developed a set of Jupyter notebooks [76] that provides a detailed explanation of each step that must be executed to use FrImCla). More advanced users can integrate FrImCla in their own programs as a standalone library. For all users, documentation is available in the

project web page explaining the different parts of the framework, the dependencies of the framework and so on.

FrImCla offers several feature extractor methods (employing both traditional and transfer-learning techniques) to describe the dataset of images. In the case of traditional Computer Vision feature extractors, FrImCla includes LAB and HSV histograms [77], Haralick features [78] and HOG features [79] — this functionality has been implemented using the OpenCV library. In the case of transfer learning methods, we employ the output from the last layer of several deep learning networks, trained for the ImageNet challenge [35], as feature extractors; namely, the user can select among VGG16 [80], VGG19 [80], DenseNet [81], ResNet [12], Inception [82], GoogleNet [83], Overfeat [84] and Xception [85] — moreover, FrImCla has been designed to easily include other feature extraction methods. The features extracted from images are used to train the classification models.

Similarly to the case of feature extraction methods, FrImCla users can select among several supervised learning algorithms including SVM [86], KNN [87], Neural networks [88], Gradient Boost [89], Logistic Regression [90], Random Forest [91] and Extremely Randomised Trees [92] for single-label datasets. In the case of multi-label datasets (that is, datasets where the same image might have several labels), FrImCla users can employ all the aforementioned algorithms through the binary relevance [93], the classifier chain [94], and the label powerset [73] methods; and, additionally, the ML-KNN [95] and the MLTSVM [96] methods are provided. In both cases, the list of algorithms can be easily extended to incorporate other techniques. The Machine Learning algorithms are trained using a stratified 10-fold cross validation, and the user can also choose among several metrics (including accuracy, F1-score, recall, precision and AUC) to measure the performance of the models. The information obtained from the cross-validation is employed to select the best combination of feature extractor and classification algorithm using a statistical analysis.

The 10-fold cross validation procedure explained previously is enough for selecting the overall best combination of feature extractor and Machine Learning algorithm. In addition, FrImCla provides a statistical method, that does not produce any computational overhead, to find whether there are significant differences among the constructed models. This procedure is widely employed in several fields [97].

FrImCla has two output modes: best classifier and ensemble of models. In the former, once the results of the statistical analysis are obtained, FrImCla analyzes which one is the best combination of feature extractor and classification algorithm. From that combination, the algorithm is retrained with the whole dataset to be further used. FrImCla generates the documentation about the choice of the best model using the accuracy or the measure selected for the training process, and it also informs the user about the time taken throughout the execution. In the latter mode, the best algorithm for each feature extractor is retrained, and it is used as a component of an ensemble based on the majority voting scheme [98] — this technique uses all the models to predict the classes of the images; and, the class with the majority of the votes is the one that results from the prediction. Normally, the ensemble mode gives better results than using just one model for prediction [98]; however, it takes more time because it uses several models to obtain the final result.

The models generated by FrImCla framework can be used in several ways. For

expert users, they can use the command line or the Python functions provided for such an aim. These users can also exploit the models within their own libraries. Another option to interact with the generated models is by using Jupyter notebooks. With this option the user can read and learn what the models need to be used. The last option is a simple web application. This option is focused on the users that only want to predict the class of their images and do not want to bother about the peculiarities of the model.

A case study: MIAS

In order to test the suitability of FrImCla, we considered four different scenarios (MIAS dataset, Malaria parasite dataset, single-label datasets and multi-label datasets). In this memoir we only include the study of the MIAS dataset, the complete study is available at [63]. The MIAS dataset consists of a set of 161 pairs of mammographic images [99]. The photos are in grayscale and they have an average size of 150×150 . The images can be classified into two classes (normal or abnormal) or they can be classified into three classes (normal, benign or malignant) depending on the severity of the abnormality. There are 113 abnormal images and 209 normal. The dataset is divided into 80% for training and 20% for testing to know the performance of the framework.

Using FrImCla, we have performed a thorough study by combining all the available feature extractors and Machine Learning models. Namely, we employed as featured extractors: VGG16, VGG19, Resnet, Inception, GoogleNet, Overfeat, DenseNet, LAB444, LAB888, HSV444 and HSV888 (where LABXYZ and HSVXYZ are respectively LAB and HSV histograms using X,Y,Z bins per channel); and as Machine Learning algorithms: SVM, KNN, Multilayer perceptron (MLP), Gradient Boost (GB), Logistic Regression (LR), Random Forest (RF), and Extremely Randomised Trees (ET).

In order to compare all the possible combinations, we analyzed the statistical study performed by FrImCla. The analysis has two steps, the former serves to determine the best Machine Learning algorithm for each feature extractor; whereas, the latter determines which is the best overall combination. From the first analysis, see Table 2.2, we can observe that using transfer learning features, we can easily obtain an accuracy higher than 80% but only with a few of them we can achieve an accuracy over 90%. On the contrary, models trained using traditional features are far from the 80% accuracy. This shows the importance of trying different models. In the second analysis, we compared the best model for each feature extractor, and we check whether the three conditions (independence, normality and heteroscedasticity) are fulfilled. As the conditions were not fulfilled, a Friedman test was performed and gave us a ranking of the compared models assuming as null hypothesis that all the models have the same performance. We obtained differences ($F = 11.36$; $p = 6.97 \times 10^{-9}$) among the models, with a large size effect $\eta^2 = 0.593969$. As a result of the analysis, we see that the best combination of feature extractor and Machine Learning algorithm was *Overfeat* with *Logistic Regression* (although similar results are obtained with other methods) with an accuracy of 93.6%. The result with the test set is a 92.53% accuracy and a 90.95% AUC.

We compared our results with the state-of-the-art models for the MIAS dataset. The best model in the literature for the MIAS dataset was presented in [100] and achieved an accuracy of 98% using fine-tuning. In [100], they fine-tuned the networks VGG16, Resnet50 and Inception v3 for the MIAS dataset; but, instead of applying transfer

Table 2.2: Mean accuracy (and standard deviation) of the different studied models for the MIAS dataset. The best result for each model in *italics*, the best result in **bold** face. ** $p < 0.01$; *** $p < 0.001$; $>$: there are significant differences; \simeq : there are not significant differences.

Network	MLP	SVM	KNN	LR	GB	RF	ET	Test (ANOVA or Fried- man)	After post-hoc procedure
Densenet	77.3 (1.3)	82.0 (7.2)	77.7 (9.8)	83.6 (5.4)	81.2 (9.6)	82.4 (1.0)	<i>86.0</i> (9.1)	0.43	ET \simeq MLP, SVM, KNN, LR, GB, RF
GoogleNet	74.8 (2.8)	84.5 (5.3)	82.8 (4.5)	<i>88.4</i> (2.8)	84.4 (5.1)	86.0 (2.7)	86.0 (4.5)	4.37**	LR \simeq SVM, KNN, LR, GB, RF; LR $\not\simeq$ MLP
Inception v3	78.8 (2.6)	85.6 (2.3)	75.3 (6.6)	85.6 (3.4)	<i>87.6</i> (2.9)	84.0 (5.6)	87.2 (4.6)	4.59**	GB \simeq SVM, LR, ET, RF; GB $\not\simeq$ KNN, MLP
Overfeat	88.4 (5.7)	93.2 (2.6)	82.4 (4.1)	93.6 (3.8)	87.2 (5.7)	87.6 (3.2)	88.4 (4.6)	2.95*	LR \simeq MLP, SVM, ET, GB, RF; LR $\not\simeq$ KNN
Resnet 50	83.6 (1.9)	84.0 (2.7)	80.4 (2.2)	85.2 (3.3)	86.9 (5.7)	88.4 (1.8)	89.2 (6.5)	2.42	ET \simeq MLP, SVM, KNN, LR, GB, RF
VGG16	82.4 (4.5)	84.4 (2.8)	82.4 (4.6)	83.6 (4.8)	82.4 (6.8)	83.2 (1.0)	<i>86.8</i> (4.1)	0.51	ET \simeq MLP, SVM, KNN, LR, GB, RF
VGG19	82.4 (5.1)	87.2 (2.8)	83.6 (3.6)	88.0 (2.4)	86.4 (5.8)	84.4 (2.8)	<i>88.8</i> (3.8)	1.44	ET \simeq MLP, SVM, KNN, LR, GB, RF
LAB444	69.3 (4.0)	68.9 (5.9)	70.5 (7.0)	69.7 (5.1)	68.1 (8.8)	68.9 (7.4)	<i>70.9</i> (8.5)	0.07	ET \simeq MLP, SVM, KNN, LR, GB, RF
LAB888	70.1 (5.6)	72.0 (7.5)	68.9 (9.1)	72.1 (6.4)	69.3 (8.2)	<i>74.0</i> (9.7)	72.9 (5.0)	0.26	RF \simeq MLP, SVM, KNN, LR, GB, ET
HSV444	70.0 (7.6)	71.7 (4.8)	69.3 (6.4)	72.5 (7.3)	69.3 (6.7)	70.1 (6.8)	67.7 (4.1)	0.18	SVM \simeq MLP, LR, KNN, RF, GB, ET
HSV888	72.1 (6.0)	<i>73.6</i> (7.4)	71.7 (9.6)	73.6 (8.0)	70.5 (8.1)	68.9 (11.1)	73.6 (8.3)	0.24	LR \simeq MLP, SVM, KNN, RF, GB, ET

learning from natural images, they fine-tuned the networks in other mammographical datasets. This shows the importance of transferring the knowledge from closer domains. However, such an approach requires networks trained in similar datasets with more than 6100 images and the use of GPUs, two restrictions that are not always fulfilled.

Up to the best of our knowledge, the best model for the MIAS dataset built only from images of this dataset was presented in [101], and also used transfer-learning from natural images, as in our case. In [101], the networks Inception, Xception and MobileNet were fine-tuned achieving an accuracy of 90% in the test set using the same percentage split employed in our work. Such an accuracy is lower than ours, and was obtained using GPUs.

Comparison with other AutoML tools

We finish by comparing the performance of FrImCla regarding other AutoML tools. Namely we compare FrImCla with AutoKeras [24], Devol [102], Ludwig [103], and WND-CHARM [104] by using 11 datasets from different types of image classification problems (these datasets were studied in [105]). The accuracy achieved by each tool is presented in Table 2.3. The execution environment used for the tests has been Google Colab¹ with the GPU option activated for AutoKeras and Devol; and for FrImCla, Ludwig and WND-CHARM, the experiments were run in a computer under Linux OS on a machine with CPU Intel Core i5-8250U 3.60GHz and 7.7 GiB of RAM.

¹<https://colab.research.google.com/>

As we can see from Table 2.3, FrImCla obtained better results in most datasets, and in several cases the improvement was very noticeable. There are only three datasets where other tools achieved a better accuracy, and even in those cases, the best accuracy and FrImCla accuracy were close. It is worth noting that in most datasets, AutoKeras, Devol, and Ludwig tended to overfit, whereas the models constructed by FrImCla and WND-CHARM generalized properly.

Table 2.3: Comparison of the performance of AutoKeras, Devol, FrImCla, Ludwig, and WND-CHARM for constructing image classification models in 11 image classification problems. The best results are highlighted in bold face

	Binucleate	C. elegans	Cho	Hela	Liver Aging	Liver Gender (AL)	Liver Gender (CR)	Lymphoma	Pollen	RNAi	Terminal
FrImCla	1.00	0.85	0.98	0.82	0.87	0.98	1.00	0.86	0.96	0.69	0.59
AutoKeras	0.55	0.66	0.96	0.47	0.91	0.98	1.00	0.89	0.81	0.24	0.47
Devol	0.54	0.69	0.75	0.68	0.43	0.82	1.00	0.55	0.89	0.28	0.36
Ludwig	0.54	0.48	0.64	0.51	0.33	0.65	0.93	0.57	0.58	0	0.53
WndCharm	1.00	0.7	0.95	0.88	0.93	0.98	0.97	0.79	0.96	0.66	0.45

Conclusions

The first contribution of this work is FrImCla, an open-source framework that allows users to easily create image classification models. This is achieved by automating all the steps required to train an image classification model. Namely, FrImCla is able to select the best combination of feature extractor and classification algorithm by conducting a comprehensive statistical study. FrImCla can be framed in the context of AutoML tools, but it has several advantages with respect to the existing tools. First of all, FrImCla automatizes the whole pipeline to construct classification models from raw images. In addition, it reduces the amount of data and computational resources that are required to train those classification models thanks to the use of transfer learning. Furthermore, the accuracy achieved by the models constructed with FrImCla is superior to the accuracy obtained by using other AutoML tools. In fact, FrImCla models can achieve close results to state-of-the-art models specific for concrete problems by using a general method that can be applied to any dataset. This work has inspired us to create another AutoML tool for object detection.

2.1.2 UFOD: An AutoML tool for object detection

Similar to image classification, Deep Learning methods for object detection have also received a lot of attention in the last few years [5]. This is due to their applications in contexts such as agriculture, manufacturing, robotics or medicine [5]. Deep Learning methods for object detection can be classified into two groups: *two-stage algorithms*, such as Faster R-CNN [15] or FPN [106], that include a step for generating object proposals, and another step for classifying such proposals; and, *one-stage algorithms*, such as RetinaNet [107], SSD [108] or YOLO [14], that conduct the detection process without the step that generates object proposals. In both kinds of algorithms, there is a trade-off between accuracy and speed; in general, one-stage detectors are faster but less accurate than two-stage detectors; and, therefore, they are applied in different contexts.

As we have mentioned in the introduction, the adoption of Deep Learning techniques for object detection by users outside the Machine Learning community is a slow process due to several factors. In addition to the challenges faced in general by Deep Learning models [109], the constant flow of new Deep Learning architectures for object detection makes it difficult to keep track of the best methods. In addition, even if there is a trend of publishing the source code of the algorithms associated with research papers, most algorithms are not prepared to train models with custom datasets, and it might be challenging to use such a code in new projects. This issue has been faced with the development of frameworks, such as the Tensorflow detection API [110], MXNet [111] or IceVision [112], that provide several Deep Learning detection algorithms ready to be trained with the users datasets.

The implementation of several algorithms in the same framework is an important feature since there is not a silver bullet solution to solve all the detection problems [16]; and, hence, it is necessary to search for the most suitable algorithm and configuration of hyperparameters for each particular problem. However, there is not a single framework that provides all the existing detection algorithms, and each library uses its own format for encoding datasets, and has its own training protocol, performance measures, and preprocessing steps. Therefore, it is difficult to train and compare different methods implemented in different frameworks, and conclude which one is the best for a particular problem. We have tackled these challenges by developing UFOD, an automated Machine Learning framework that aims to facilitate the construction and usage of Deep Learning object detectors.

As in the case of FrImCla, UFOD is aligned with systems such as Auto-Sklearn or Auto-Weka; that is, our framework is designed to help non-expert users by automatically searching through the space of object detection algorithms (available in several frameworks and libraries) to maximize their performance. This approach differs to most AutoML methods for object detection since they are based on NAS methods [22], but they suffer the same drawbacks mentioned in the case of NAS techniques for image classification. In the rest of this section, we present the most relevant features of UFOD.

UFOD description

UFOD is an open-source library that has been implemented in Python, and has been designed to simplify and automatize the process of training multiple object detection models using different libraries, and select the best of them.

We have designed a workflow that captures all the necessary steps to train several object detection models for custom datasets and select the best one. Such a workflow can be summarized as follows, see Figure 2.2. First of all, the user selects the dataset of images and some configuration parameters (mainly, the algorithms that will be trained and the metric to evaluate them) — this information is provided by means of a JSON file, see Figure 2.3. Subsequently, the dataset is split into a training set and a testing set (if this division is not explicitly provided). The training set is employed to construct several object detection models using fine-tuning, and the best of those models is selected based on their performance on the testing set; if several models achieve the same performance, the fastest model among them is returned. The output of the framework is either the best model, or an ensemble of the models. In addition, an application, in the form of a

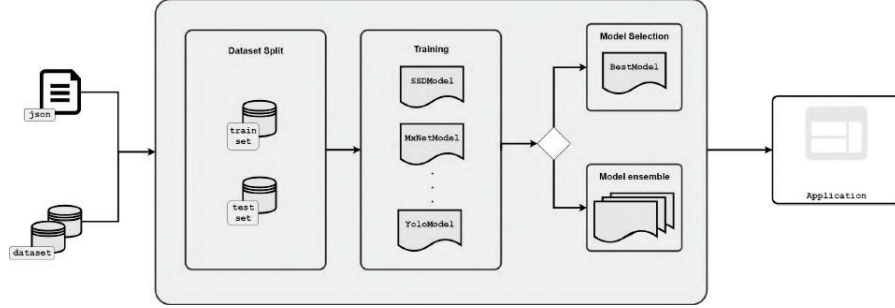


Figure 2.2: Workflow of the framework

```
{
  "dataset" : "../datasets/ReconyxCameraTrapData",
  "dataset_name": "ReconyxCameraTrapData",
  "exec" : {"type":"local", "ngpus": 2},
  "frameworks" : [{"Mxnet", "ssdVgg16"}, {"Mxnet", "ssdVgg16_512"}, {"Mxnet", "ssdResnet"},
    {"Mxnet", "ssdMobilenet"}, {"Rcnn", "mask-rcnn"}, {"Darknet", "yolo"}, {"Darknet", "tinyYolo"}]
}
```

```
> python3 taskLauncher.py -c config.json
```

Figure 2.3: **Top.** Example of a configuration file for UFOD. **Bottom.** Command to start the training process.

Jupyter notebook is provided to employ such a model or ensemble of models. Apart from the first step — that is, the selection of the models to be trained — the rest of the process is conducted automatically without any user intervention in our framework.

Since, the final aim of UFOD is the construction of a unified framework for object detection as complete as possible and that can be easily extensible with new algorithms, we have designed a high-level API that allows the integration of object detection algorithms independently of the framework or underlying library employed to implement them. The API has been designed to easily include new algorithms independently of their underlying library. Currently UFOD supports the following object detection algorithms YOLO [14], TinyYolo [14] (Darknet [14]), EfficientDet [113] (Keras [114]), FCOS [115] (FCOS Keras [115]), FSAF [116] (FSAF Keras [116]), MaskRCNN [117] (Mask RCNN Keras [118]), SSD [108] (MxNet [119] and Tensorflow [120]), Faster RCNN [15] (MMDetection [121] and Tensorflow), Cascade RCNN [122] (MMDetection), Retinanet [107] (Retinanet Keras and MMDetection) and RFCN [123] (TensorFlow).

One of the challenges that we faced to integrate those algorithms was the particularities of each library; and, namely the different formats employed by them. To deal with this issue, UFOD's high-level API includes the necessary methods to automatically transform the provided images and annotations (in the Pascal VOC format [124]) to the format required for each detection algorithm, and also sets the environment to train the algorithm.

Once the environment for training each algorithms is set, the training process can

start by applying transfer learning to the selected algorithms. UFOD can be run locally (in a single computer with a GPU), using a cloud environment like Google Colab, or in a cluster of computers. The reason to include the cluster execution mode is the fact that training object detection algorithms is a time-consuming task. Therefore, if several models must be trained, this task can take advantage of a distributed process to train them at the same time — this feature is implemented using the cluster manager SLURM [125]. Depending on the execution mode, some additional parameters must be configured; for example, the number of available GPUs if working locally, or the time that the tasks can be run in the cluster.

After training the object detection models, it remains the question of deciding which model produces the best results. Each underlying library included in UFOD can compute the performance of their own models; however, different frameworks employ different evaluation metrics; and, therefore, it is difficult to compare the models produced with different tools. To deal with this problem, we have included a two-step procedure to evaluate algorithms independently of the underlying framework. Namely, given a folder with the images and annotations that will be employed for testing a model, (1) the model detects the objects in those images and stores the result in the Pascal VOC format; and (2) the original annotations and the generated detections are compared with a particular metric. Currently, metrics such as mAP, IoU or F1-score are supported. Using this approach, the models constructed using our framework can be compared even if they were constructed using different tools, and the comparison is returned to the user both textually and visually. When two models obtain the same accuracy with respect to the selected metric, the best model is chosen based on its speed for prediction (either using a GPU or a CPU).

Finally, the last key feature of UFOD is its output, that can be either the best model selected as indicated previously, or an ensemble of models that combines either all the trained models or the best three models using the Non-Maximum Suppression algorithm [126]. In both cases, the necessary code to run the models is also generated, and it can be called from the command line, integrated in other Python programs, or executed by using Jupyter notebooks.

A case study: the Optic dataset

In order to test the performance of the UFOD framework, we present the results obtained by using UFOD with the Optic dataset [127]. The complete study with another dataset is available in [66]. The Optic dataset consists of ophthalmological images that are employed to detect discs where blood vessels join together inside the eyeball. This dataset contains 1250 images, and there is only one disc per image. The Optic dataset is small in comparison with the Pascal VOC or COCO datasets, but such a size is common when training a detection algorithm in a custom dataset.

For the experiment, we employed the following settings. The dataset was split into two different sets using a 75% for training, and a 25% for testing. The training set was employed to construct models using all the algorithms available in our framework. The performance of the trained models was evaluated on the test set using three different metrics: F1-score, IoU, and mAP. Once the user has fixed the training options in the configuration file, the training process was launched by just executing an instruction in

Table 2.4: Results using different models for the Optic dataset. The best result for the individual models is shown in italics, and the best overall result is shown in bold face.

	F1	IoU	mAP
Cascade RCNN-50	100	91.39	90.91
EfficientDet-B0	100	90.97	90.91
Faster-RCNN	99.49	90.09	90.91
FCOS-Resnet-50	100	89.02	100
FSAF-Resnet-50	16.12	7.44	6.44
MRCNN-50	94.91	72.65	90.91
MRCNN-101	87.87	71.07	90.84
Retinanet-50	99.49	90.40	90.91
Retinanet-101	99.49	89.42	100
Retinanet-152	99.49	89.66	100
RFCN	97.43	87.69	90.84
SSD VGG16-300	98.99	84.66	90.91
SSD VGG16-512	97.44	83.35	90.91
SSD Mobilenet (MXNET)	99.63	80.60	90.88
SSD Mobilenet (Tensorflow)	97.95	87.22	90.84
SSD Resnet	99.00	86.77	90.91
Tiny YOLO	87.01	60.79	90.67
YOLO	91.89	75.56	90.91
Ensemble	99.63	80.60	90.91
Ensemble 3 best	100	89.61	100

the command line, see Figure 2.3.

The results for the Optic dataset are included in Table 2.4. This dataset can be seen as a sanity check for our framework since good models can be constructed using all the algorithms provided in our framework. Namely, we obtained a mAP of 100% with 3 models, and the rest of the models, but the FSAF model, achieved a mAP over 90%. There are differences among the models when using the F1-score and the IoU metrics, but accurate results were obtained with all of them. In addition, we improved those models using the ensemble process.

Conclusions

The second contribution of this work is UFOD, a free and open-source framework that aims to facilitate the construction and use of the most suitable object detection model for the particular problem of each user. Namely, our framework allows users to easily train, compare and ensemble several object detection models on their own datasets using a common pipeline. The framework can be seen as a wrapper for the functionality provided by the multiple and heterogeneous existing frameworks and libraries that support object detection algorithms. This is achieved thanks to a high-level extensible API that abstracts all the steps that are required to train and evaluate a detection model. It is important to note that, as far as we are aware, UFOD is the first AutoML tool for object detection.

Both UFOD and FrImCla can be used as an inspiration, or as a starting point, to create AutoML tools for other Computer Vision tasks such as semantic segmentation or anomaly detection, but this remains future work. For both FrImCla and UFOD, and also for other AutoML tools developed in the future, the more images feed to them the

merrier since Deep Learning algorithms are data hungry. Therefore, we have developed a tool with the aim of generating large enough datasets.

2.2 Objective 2: A General Image Augmentation Library

Deep Learning methods are data demanding, which is a problem in context such as biology, medicine or agriculture where it is challenging to acquire new images (for instance due to a tight budget or necessity of an invasive procedure) [8, 128, 129]. Moreover, once the images have been acquired, they must be manually annotated, a task that is time-consuming and requires experts in the field to conduct it correctly [130].

A successful method that has been applied to deal with the problem of a limited amount of data is data augmentation [9, 39]. This technique consists in generating new training samples from the original dataset by applying transformations that do not alter the class of the data. This method has been successfully applied in several contexts such as brain electron microscopy image segmentation [131], melanoma detection [128], or the detection of gastrointestinal diseases from endoscopic images [8]. Due to this fact, several libraries, like Augmentor [41] or Imgaug [40], and Deep Learning frameworks, like Keras [36] or Tensorflow [120], provide features for data augmentation in the context of image classification.

In general, those augmentation libraries have not been designed to deal with other common tasks in Computer Vision such as object localization, object detection, semantic segmentation or instance segmentation. These problems can also take advantage from data augmentation [131, 132]; but, at least up to the best of our knowledge, it does not exist a general purpose library that can be applied to those problems and works with the standard annotation formats. This is probably due to the fact that, in the classification context, transformation techniques for image augmentation do not generally change the class of an image, but they might alter the annotation in the other problems. For instance, applying the vertical flip operation to a melanoma image does not change the class of the image; but the position of the melanoma in the new image has changed from the original image. This means that, for each specific problem, special purpose methods must be implemented, or artificially generated images must be manually annotated. Neither of these two solutions is feasible when dealing with hundreds or thousands of images. In addition, augmentation libraries focus on datasets of 2 dimensional (2D) images, but do not deal with multi-dimensional images (such as z-stacks or videos).

To tackle the aforementioned challenges, we have designed a generic method, that can be applied to automatically augment a dataset of images devoted to classification, localization, detection, semantic segmentation, and instance segmentation using the classical image augmentation transformations applied in object recognition; moreover, this method can be also applied to multi-dimensional images. Such a method has been implemented in an open-source library called CLoDSA (that stands for Classification, Localization, Detection, Segmentation Augmentor) [68].



Figure 2.4: Examples of annotations for classification, localization, detection and semantic segmentation.

CLoDSA method

We have developed an approach to augment images for the problems of object classification, localization, detection, semantic segmentation and instance segmentation. First of all, it is important to understand how the images are annotated in each of these five problems. In the case of object classification, each image is labeled with a prefixed category; for object localization, the position of the object in the image is provided using the bounding box (that is, the minimum rectangle containing the object); for object detection, a list of bounding boxes and the category of the objects inside those boxes are given; in semantic segmentation, each pixel of the image is labeled with the class of its enclosing object; and, finally in instance segmentation, each pixel of the image is labeled with the class of its enclosing object and objects of the same class are distinguished among them. An example of each kind of annotation is provided in Figure 2.4. It is worth noting that, instance segmentation is the most general case, and the other problems can be seen as particular cases of such a problem; however, special purpose techniques and annotation formats have been developed to tackle each problem; and, therefore, we consider them separately. Image augmentation for object classification is the simplest case. This task consists in specifying a set of transformations for which an image classification problem is believed to be invariant; that is, transformations that do not change the class of the image. It is important to notice that image-augmentation techniques are problem-dependent and some transformations should not be applied; for example, applying a 180° rotation to an image of the digit “6” changes its class to the digit “9”.

In the literature, the most commonly chosen image augmentation techniques for object classification are geometric transformations (such as translations, rotations, or scaling), color transformations (for instance, changing the color palette of the image or normalizing the image), filters (for example, Gaussian or median filters), and elastic distortions [39]. Other more specific techniques such as Generative Adversarial Networks (GANs) [44] have been also applied for image augmentation in object classification [46]; however, we will not consider GANs in our work since they cannot be directly applied for image augmentation in the other four problems.

For image augmentation in localization, detection, segmentation, and instance segmentation, we consider the classical image augmentation techniques applied in

Table 2.5: List of considered augmentation techniques

Position invariant techniques	Position variant techniques
Average blur	Crop
Bilateral blur	Elastic deformation
Brightness noising	Flip
Color noising	Rescale
Contrast noising	Rotation
Dropout	Skewing
Gamma correction	Translation
Gaussian blur	
Gaussian noise	
Hue jitter	
Median blur	
Normalization	
Random erasing	
Salt and pepper	
Saturation jitter	
Sharpen	
Value jitter	
Channel shift	
Lightning	
Change space color	

object classification, and split them into two categories. The former category consists of the techniques that leave invariant the position of the objects in the image; for example, changing the color palette of the image does not modify the position of an object. On the contrary, techniques that modify the position of the image belong to the latter category; for instance, rotation and translation belong to this category. A list of all the transformations that have been considered in this work, and their corresponding category, is available in Table 2.5.

Image augmentation for localization, detection, segmentation, and instance segmentation using the techniques from the “invariant” category consists in applying the technique to the image and returning the resulting image and the original annotation as result. The rest of this section is devoted to explain, for each problem, how the annotation can be automatically generated for the techniques of the “variant” category.

In the case of object localization, the first step to automatically generate the label from an annotated image consists in generating a mask from the annotated bounding box — i.e. a black image with a white rectangle indicating the position of the object. Subsequently, the transformation technique is applied to both the original image and the generated mask. Afterwards, from the transformed mask, the white region is simply located using basic contours properties, and the bounding box of the region is obtained — some transformations might generate a really small bounding box, or produce an image without bounding box at all since it will be located outside the boundaries of the image; to avoid that problem, a minimum percentage is required to keep the image; otherwise, the image is discarded. Finally, the transformed image is combined with the resulting bounding box to obtain the new annotated image. This process is depicted in Figure 2.5 using as example the horizontal flip operation.

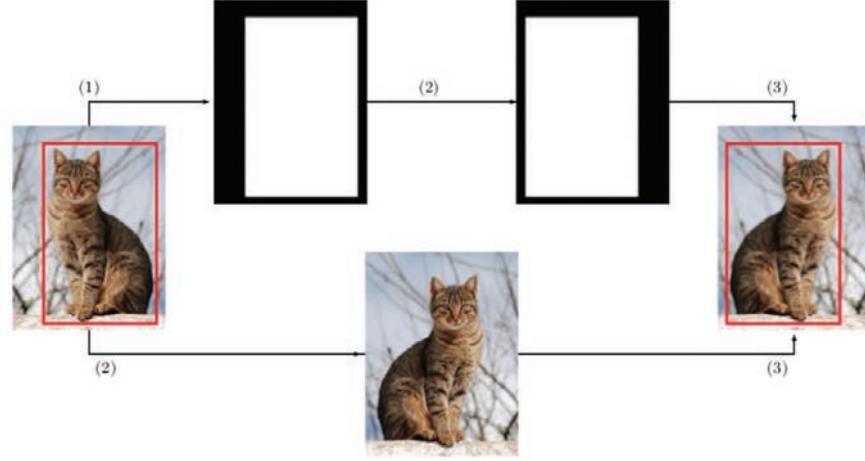


Figure 2.5: Process to automatically label augmented images for the localization problem: (1) generation of the mask, (2) application of the transformation operation (horizontal flip) to both the mask and the original image, and (3) combination of the bounding box containing the new mask and the transformed image

The procedure for image augmentation in object detection relies on the method explained for object localization. Namely, the only difference is that instead of generating a unique mask, a list of masks is generated for each bounding box of the list of annotations.

In the semantic segmentation problem, given an image I , each pixel $I(i,j)$ of the image — i.e. the pixel of row i and column j of I — is labeled with the class of its enclosing object, this annotation is usually provided by means of an image A of the same size as the original image, where $A(i,j)$ provides the category of the pixel $I(i,j)$, and where each pixel category is given by a different value. In this case, the idea to automatically generate a new annotated image consists in applying the same transformation to the original and the annotation image, the result will be the combination of the two transformed images.

Finally, we present the procedure for the instance segmentation problem. The idea is similar to the method explained for object detection. A mask is generated for each instance of the image. Subsequently, the transformation technique is applied to both the original image and the generated masks. Afterwards, from the transformed masks, the new instances are obtained.

The aforementioned procedures are focused on 2D images, but they can also be applied to multi-dimensional images that can be decomposed as a collection of images — this includes z-stacks and videos among others. The method consists in decomposing the multi-dimensional image into a collection of 2D images, applying the corresponding procedure, and finally combining back the resulting images into a multi-dimensional image. Both methods for 2D and multi-dimensional images have been implemented in a library called CLoDSA.

CLoDSA description

CLoDSA is an open-source library implemented in Python and relies on OpenCV [74] and SciPy [133] to deal with the different augmentation techniques. The CLoDSA library can be used in any operating system, and it is also independent from any particular Machine Learning framework.

CLoDSA augmentation procedure is flexible to adapt to different needs and it is based on six parameters: the dataset of images, the kind of problem, the input annotation mode, the output annotation mode, the generation mode, and the techniques to be applied. The dataset of images is given by the path where the images are located; and the kind of problem is either classification, localization, detection, segmentation, instance segmentation, stack classification, stack detection, or stack segmentation (the former five can be applied to datasets of 2D images, and the latter 3 to datasets of multi-dimensional images). The other four parameters, and how they are managed in CLoDSA, deserve a more detailed explanation.

The input annotation mode refers to the way of providing the labels for the images. CLoDSA supports the most-widely employed formats for annotating classification, localization, detection, semantic and instance segmentation tasks. For example, for object classification problems, the images can be organized by folders, and the label of an image be given by the name of the containing folder; another option for object classification labels is a spreadsheet with two columns that provide, respectively, the path of the image and the label; for object localization and detection there are several formats to annotate images such as the PASCAL VOC format [124] or the OpenCV format [74]; for semantic segmentation, the annotation images can be given in a devoted folder or in the same folder as the images; and, for instance segmentation, the COCO format is usually employed [29]. CLoDSA has been designed to manage different alternatives for the different problems, and can be easily extended to include new input modes that might appear in the future. To this aim, several design patterns, like the Factory pattern [134], and software engineering principles, such as dependency inversion or open/closed [135], have been applied. The list of input formats supported by CLoDSA is given in Table 2.6 — a detailed explanation of the process to include new formats is provided in the project webpage.

The output annotation mode indicates the way of storing the augmented images and their annotations. The first option can be as simple as using the same format or approach used to input the annotations. However, this might have the drawback of storing a large amount of images in the hard drive. To deal with this problem, it can be useful to store the augmented dataset using the standard Hierarchical Data Format (HDF5) [136] — a format designed to store and organize large amounts of data. Another approach to tackle the storage problem, and since the final aim of data augmentation is the use of the augmented images to train a model, consists in directly feeding the augmented images as batches to the model, as done for instance in Keras [36]. CLoDSA features these three approaches, and has been designed to easily include new methods in the future. The complete list of output formats supported by CLoDSA is given in Table 2.6.

The generation mode indicates how the augmentation techniques will be applied. Currently, there are only two possible modes: linear and power — in the future, new modes can be included. In the linear mode, given a dataset of n images, and a list

Table 2.6: List of supported annotation formats

Data	Problem	Input format	Output format
2D Images	Classification	A folder for each class of image	A folder for each class of image An HDF5 file A Keras generator Pascal VOC format An HDF5
	Localization	Pascal VOC format	Pascal VOC format An HDF5
	Detection	Pascal VOC format YOLO format	Pascal VOC format YOLO format
	Segmentation	A folder containing the images and their associated masks	A folder containing the images and their associated masks An HDF5 file A Keras generator COCO format
	Instance segmentation	COCO format JSON format from ImageJ	JSON format from ImageJ
Multi-dimensional Images	Video Classification	A folder for each class of video	A folder for each class of video
	Video Detection	Youtube BB format	Youtube BB format
	Stack segmentation	Pairs of tiff files containing the stack and the associated mask	Pairs of tiff files containing the stack and the associated mask

of m augmentation techniques, each technique is applied to the n images producing at most $n \times m$ images. The power mode is a pipeline approach where augmentation techniques are chained together. In this approach, the images produced in one step of the pipeline are added to the dataset that will be fed in the next step of the pipeline producing a total of $(2^m - 1) \times n$ new images (where n is the size of the original dataset and m is the cardinal of the set of techniques of the pipeline). Finally, the last but not least important parameter is the set of augmentation techniques to apply — the list of techniques available in CLoDSA is given in Table 2.5, and a more detailed explanation of the techniques and the parameters to configure them is provided in the CLoDSA documentation. Depending on the particular problem, the CLoDSA users can select the techniques that are more fitted for their needs.

CLoDSA can be employed by both expert and non-expert users. First of all, users that are used to work with Python libraries can import CLoDSA as any other library and use it directly in their own projects. Several examples of how the library can be imported and employed are provided in the project webpage. This kind of users can extend CLoDSA with new augmentation techniques easily. The second, and probably the most common, kind of CLoDSA’s users are researchers that know how to employ Python but do not want to integrate CLoDSA with their own code. In this case, we have provided several Jupyter notebooks to illustrate how to employ CLoDSA for data augmentation in several contexts — again the notebooks are provided in the project webpage.

CLoDSA can be also employed without any knowledge of Python. To this aim, CLoDSA can be executed as a command line program that can be configured by means of a JSON file. Therefore, users who know how to write JSON files can employ this approach. Finally, and due to the fact that the creation of a JSON file might be a challenge for some users since there is a great variety of options to configure the library; we have created a step-by-step Java wizard that guides the user in the process of creating the JSON file and invoking the CLoDSA library. In this way, the users, instead of writing a JSON file, select in a simple graphical user interface the different options for augmenting their dataset of images, and the wizard is in charge of generating the JSON file and executing the augmentation procedure. Besides, since new configuration options might appear in the future for CLoDSA, the Java wizard can include those options by modifying a configuration file — this avoids the issue of modifying the Java wizard every time that a new option is included in CLoDSA.

A case study: Malaria parasite classification

To show the benefits of applying data augmentation using CLoDSA, we considered classification of Malaria images [137] combining FrImCla and CLoDSA. In this dataset images are labeled as parasitized or uninfected; and, we analyzed the impact of applying data augmentation. Namely, we considered 7 publicly available networks trained on the ImageNet challenge (the networks are GoogleNet, Inception v3, OverFeat, Resnet 50, VGG16, VGG19, and Xception v1) and use them as feature extractors to construct classification models for the Malaria dataset. For each feature extractor network, we considered 4 datasets: D1 is the original dataset that consists of 1000 images (500 images per class); D2 was generated from D1 by applying flips and rotations (D2 consists of 5000 images, the original 1000 images and 4000 generated images); D3 was generated from D1 by applying gamma correction and equalisation of histograms (D3 consists of 3000 images, the original 1000 images and 2000 generated images); and, D4 is the combination of D2 and D3 (D4 consists of 7000 images, the original 1000 images and 6000 generated images). In order to evaluate the accuracy of the models, a stratified 5-fold cross-validation approach was employed using the FrImCla framework.

As can be seen in the scatter plot of Figure 2.6, the accuracy of the models constructed for each feature extractor method increased when data augmentation is applied. The improvement ranged from a 0.4% up to a 6.5%; and, there was only one case where applying data augmentation had a negative impact on the accuracy of the model. Moreover, we can notice that we obtained better models only applying flips and rotations (dataset D2) than using a bigger dataset where we have applied not only flips and rotations but also color transformations (dataset D4). This indicates the importance of correctly selecting the set of data augmentation techniques — an active research area [43, 138, 139].

Conclusions

The third contribution of this work is CLoDSA, an open-source framework that allows researchers to automatically apply image augmentation techniques to the problems of object classification, localization, detection, semantic segmentation, and instance segmentation. Such a method works not only with 2D images, but also with multi-dimensional images (such as stacks or videos). This library has been designed using several object oriented patterns and software engineering principles to facilitate its usability and extensibility.

Up to now, we have focused on building tools that facilitate the construction of Deep Learning models, but once those models have been trained, we want to fix the problems that appear when they are used with images from a different domain than the one used for training; that is, the domain shift challenge.

2.3 Objective 3: Dealing with the Domain Shift Problem

There is an important generalization challenge when using trained models that is known as *domain shift* (also known as distribution shift) [10, 11]. This problem arises when the data distribution of the dataset used for training a model is not the same than

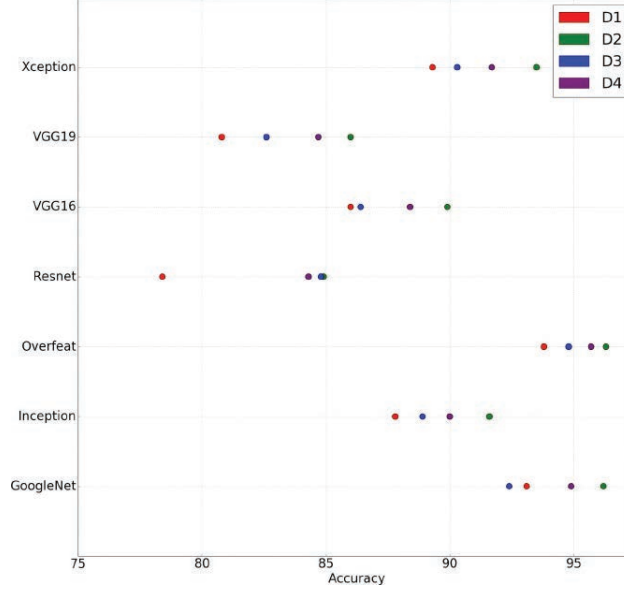


Figure 2.6: Scatter plot showing the accuracy of the models constructed for the different versions of the Malaria dataset (where D1 is the original dataset; and D2, D3 and D4 are the augmented datasets) using different feature extractor methods.

the data that the model encounters when deployed. This is common in biomedical contexts since images greatly vary due to experimental conditions, the equipment (for instance, microscopes) and settings (for instance, focus and magnification) employed for capturing those images; or in autonomous driving, where we might have different climatic conditions.

This generalization problem can be tackled by combining datasets from multiple sources [57] or by using techniques like data augmentation; nevertheless, it is not possible to foresee every new and unknown distribution. A different approach consists in applying transfer learning. However, this requires the annotation of the target dataset, a time-consuming task that should be carried out for every new dataset. A different approach to handle the domain shift problem is the application of image-to-image translation methods [48], a set of techniques that aim to learn the mapping between an input image and an output image using a training set of aligned image pairs; however, this requires paired data from the source and target domains, a challenge that can be faced by using unpaired image-to-image translation techniques [49].

Unpaired image-to-image translation methods translate images from a domain A to a domain B , and vice versa. This approach poses two challenges. First, datasets from both domains must be available, and this might be an issue due to privacy concerns [62]; and, secondly, unpaired image-to-image methods are mainly based on GANs that must be trained, a process that demands the usage of GPUs and might be challenging for several users due to the difficulties of training GAN models [47]. The approach that we

propose to tackle these drawbacks consists in developing a framework that facilitates the application of unpaired image-to-image methods for dealing with the domain shift problem, and also incorporate style transfer methods.

Methods

We start by explaining the procedure to apply unpaired image-to-image translation methods to tackle the domain shift problem, see Figure 2.7. We assume that a model has been trained using a source dataset of images, and we have a dataset of images with a different data distribution called the target dataset. From the source and target datasets, we build a model that transforms images following the data distribution of the source dataset to images following the data distribution of the target dataset, and vice versa. Now, when we are interested in obtaining the prediction associated with a target image, we first employ the transformation model to transform the image; and, subsequently, the transformed image is fed to the prediction model. In this approach, the key component is the algorithm employed to construct the transformation model. Currently, the most successful approaches for this task are based on GANs [44]; and, namely, variants of the CycleGAN algorithm [49], which translates an image from a source domain X to a target domain Y by learning two mappings $G_X : X \rightarrow Y$ and $G_Y : Y \rightarrow X$ such that they satisfy the cycle-consistency properties; that is $G_Y(G_X(x)) \approx x$ and $G_X(G_Y(y)) \approx y$.

We focus now on the procedure to apply style transfer methods to deal with the domain shift problem of a model — such a procedure is summarized in Figure 2.7. Analogously to the unpaired image-to-image approach, we assume that a model has been trained using a source dataset of images, and we are interested in applying such a model to obtain the prediction associated with an image from a different distribution than the source dataset; we call this image, the target image. Instead of feeding the target image directly to the model, we first take an image from the source dataset and transfer the style of that image to the target image but preserving its content producing as a result a transformed image. Finally, the transformed image is fed to the model to obtain the associated prediction. As in the unpaired image-to-image approach, the key component of the style transfer process is the algorithm that transfers the style from the source dataset but keeping the content of the target image; and, we can find several of them in the literature [140].

It is worth noting that both the unpaired image-to-image approach and the style transfer approach can be applied to deal with the domain shift problem for any Computer Vision task. Hence, these methods can be helpful for a great variety of scenarios. However, it might be difficult to apply these techniques since unpaired image-to-image methods and style transfer algorithms are implemented in different libraries and using different frameworks. Following the approach employed for developing UFOD, we have addressed this drawback by developing a high-level Python API that allows the integration of both kinds of algorithms independently of their underlying library and framework.

Currently, our library includes three style transfer algorithms: neural style transfer (NST) [45], an optimization technique that uses a Convolutional Neural Network (CNN) to decompose the content and style from images; deep image analogy [141], a

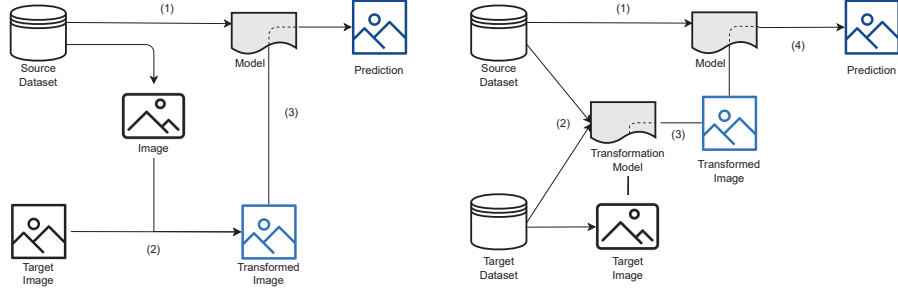


Figure 2.7: Workflow of the style transfer approach (**Left**). (1) A model is trained using a source dataset. (2) The target image is transformed using the style from an image of the source dataset. (3) The transformed image is fed to the model. Workflow of the unpaired image-to-image translation models (**Right**). (1) A model is trained using a source dataset. (2) A source dataset and a target dataset are employed to combine a GAN model. (3) Given an image from the distribution of the target dataset, the GAN model is employed to transform the image. (4) The transformed image is fed to the prediction model.

method that finds semantically-meaningful correspondences between two input images by adapting the notion of image analogy with features extracted from a CNN; and STROTSS [142], a variant of the NST algorithm that changes the optimization objective of NST. In order to apply the style transfer procedure using our API, users only have to provide the style image, the target image, and the name of the algorithm to apply; the rest of the transformation process is automatically conducted by the API.

Moreover, the library provides 6 unpaired image-to-image translation algorithms: CycleGAN [49], DualGAN [51], ForkGAN [52], GANILLA [53], CUT [50], and FASTCUT [50]); after that, the transformation model is automatically trained and the images from the target dataset are transformed. In this case, users of the API only have to provide the path to the source and target datasets, and the name of the translation algorithm to apply; after that, the transformation model is automatically trained and the images from the target dataset are transformed.

A running example

As a running example for testing our approach and library, we have considered the segmentation of spheroids. Spheroids are the most widely used 3D models to study cancer since they can be used for studying the effects of different micro-environmental characteristics on tumor behavior and for testing different preclinical and clinical treatments [143]. The images from tumor spheroids greatly vary depending on the experimental conditions, and also on the equipment (microscopes) and conditions (focus and magnification) employed to capture the images [56].

For our experiments, we have employed the 4 datasets presented in [56]; an image of each dataset is shown in Figure 2.8. As can be noticed from Figure 2.8, there are considerable differences among the images of each dataset. Three of those datasets (the BL5S, BN2S, and BN10S datasets) were employed for training 4 segmentation

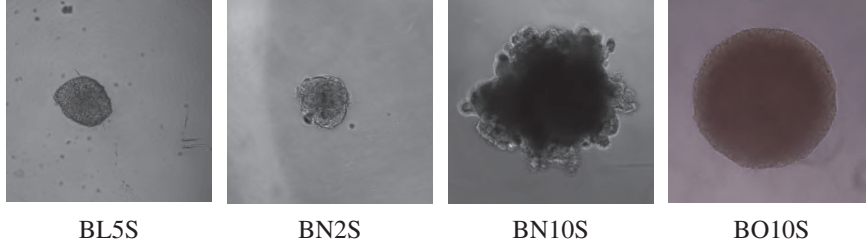


Figure 2.8: Samples from the 4 datasets employed in this work

Table 2.7: Performance of the 4 models when evaluating in a test set formed from images following the same distribution than the training set (BL5S-BN2S-BN10S), and when evaluated using a test set from a different distribution (BO10S)

	DeepLab v3	HRNet-Seg	U-Net	U2-Net
BL5S-BN2S-BN10S	97.00	97.32	97.25	97.26
BO10S	83.61	92.65	13.64	95.65

models (using the algorithms DeepLab v3 [144], HRNet Seg [145], U-net [132] and U²-Net [146]) and the last dataset (the BO10S dataset) was employed for testing. We have used this dataset split because the last dataset comes from a different laboratory so its style will not be the same as the others. The definition of those 4 architectures is available in the SemTorch package². All the architectures were trained with the libraries PyTorch [37] and fastai [38] and using a GPU Nvidia RTX 2080 Ti. In order to set the learning rate for the different architectures, we employed the procedure presented in [38]; and, we applied early stopping when training all the architectures to avoid overfitting. The metric employed to measure the accuracy of the different methods was the IoU [147]. In Table 2.7, we can see the domain shift problem with real data. In all models the results worsen when applied to images from the BOIOS dataset. These results are due to the fact that the dataset used is from a different domain than the one used in the other three datasets. The framework through the change of domain seeks to mitigate this problem and improve the results.

For training the translation models, we employed a GPU Nvidia RTX 2080 Ti to segment tumour spheroids, and using our API we randomly picked an image from the combination of the datasets BL5S, BN2S, and BN10S, and used it to transform the images from the BO10S dataset. Subsequently, we fed those images to the segmentation models, and evaluated their performance, see Table 2.8. From the three studied style transfer algorithms included in our library, both the NST and STROTSS algorithms handled the domain shift problem; whereas, the images transformed with the deep image analogy algorithm produced even worse results than the original images from the BO10S dataset. Using the NST algorithm, all the segmentation models improved their IoU (the U-Net model improves its performance from 13.64% to 89.21%, and the other

²The SemTorch package is available at <https://github.com/WaterKnight1998/SemTorch>.

Table 2.8: Performance for the BO10S dataset using the different style-transfer methods to deal with the domain shift problem. A \uparrow indicates an improvement with respect to the base model, whereas a \downarrow indicates a declination in the performance.

	DeepLab v3	HRNet-Seg	U-Net	U ² -Net
Base	83.61	92.65	13.64	95.65
NST	95.64 \uparrow	94.91 \uparrow	89.21 \uparrow	95.89 \uparrow
Deep Image Analogy	0.00 \downarrow	45.13 \downarrow	0.66 \downarrow	0.84 \downarrow
STROTSS	94.86 \uparrow	92.38 \downarrow	78.08 \uparrow	94.14 \downarrow

Table 2.9: Performance for the BO10S dataset using the different unpaired image-to-image translation methods to deal with the domain shift problem. A \uparrow indicates an improvement with respect to the base model, whereas a \downarrow indicates a declination in the performance.

	DeepLab v3	HRNet-Seg	U-Net	U2-Net
Base	83.61	92.65	13.64	95.65
CycleGAN	94.97 \uparrow	92.97 \uparrow	72.34 \uparrow	95.87 \uparrow
DualGAN	4.09 \downarrow	73.37 \downarrow	24.67 \uparrow	34.45 \downarrow
ForkGAN	32.63 \downarrow	46.10 \downarrow	38.33 \uparrow	44.46 \downarrow
GANILLA	24.27 \downarrow	76.24 \downarrow	3.26 \downarrow	82.97 \downarrow
CUT	0.48 \downarrow	38.01 \downarrow	20.94 \uparrow	52.20 \downarrow
FastCUT	6.08 \downarrow	79.52 \downarrow	1.12 \downarrow	2.98 \downarrow

models have an IoU close to 95%). For the STROTSS algorithm, the results were also positive: two of the segmentation models improved (DeepLab and U-Net), and the other two achieved worse results, but still their IoU was over 92%.

The results obtained with the unpaired image-to-image translation are summarized in Table 2.9. From the 6 algorithms included in our library, only the CycleGAN method solved the domain shift problem in our context. Using the transformation model produced by this algorithm, all the segmentation models improved their IoU (the U-Net model improves its performance from 13.64% to 72.34%, and the other models have an IoU over to 92%). On the contrary, the images produced by the rest of the transformations models were more difficult to segment, and the performance of the segmentation models decreased — the exception was the U-Net model that, in some cases, obtained better results with the transformed images, but still its IoU was under 40%.

We can visually inspect the images produced by the different transformation algorithms to discover the difficulties faced by the segmentation models, see Figure 2.9. We can notice that the three successful models (NST, STROTSS, and CycleGAN) produced images that preserved the content of the image but with a style that is similar to the style of those used for training the segmentation models. On the contrary, the deep image analogy method, and the DualGAN and ForkGAN models did not keep the content of the image; and, thus the segmentation models were not able to properly segment























































	Image	Truth	DeepLab v3	HRNet Seg	U-Net	U ² -Net
Base						
NST						
Deep Image Analogy						
STROTSS						
CycleGAN						
DualGAN						
ForkGAN						
GANILLA						
CUT						
FastCUT						

Figure 2.9: An example showing the segmentation produced by the DeepLab, HRNet, U-Net and U²-Net models after applying a style transfer algorithm or an unpaired image-to-image translation model to a given image.

the images. For the rest of the image-to-image transformation models (GANILLA, CUT and FastCUT), the content of the image was kept, but the style was not properly transferred (colour artefacts are added to the transformed image); hence, the images were not segmented properly by the models.

Conclusions

The forth contribution of this work is a framework that addresses the domain shift problem by giving access to style transfer and unpaired image-to-image algorithms. Using this framework; we have shown that using those translation methods, it is possible to recover the performance of a model that suffers from the domain shift problem. In addition, we have shown that style transfer methods achieve similar results to those obtained by unpaired image-to-image translation methods with the advantage of not requiring a training step, and therefore can be deployed by providing a single image from the source dataset. This is a relevant result since these methods do not require the training step of image-to-image translation models, can be run in a computer without special purpose hardware like GPUs, and only require the availability of an image from

the source dataset. We have also noticed that style transfer techniques and image-to-image translation methods have a different impact on the performance of the models; hence, it is important to have a simple approach to test different algorithms. This has been solved in this work with the development of a high-level API that facilitates the process of testing different alternatives for style transfer and unpaired image-to-image translation.

Up to now, we have developed several tools that facilitate the construction and use of Deep Learning models. In the rest of this memoir, we focus on two concrete biomedical problems that have been tackled using techniques implemented in our tools. In addition, the solutions given to those problems have provided us valuable experience to improve in the future our tools with methods and techniques that work in real-life problems.

2.4 Objective 4: Applications to biomedical problems

In the rest of this chapter, we present how the techniques and methods developed in the previous section have been the basis to tackle two biomedical problems that are the prediction of epiretinal membrane from retinal fundus images and the classification and segmentation of bacteria in motility images.

2.4.1 Prediction of ERM from Retinal Fundus Images

An epiretinal membrane (ERM) is a fibrocellular tissue found on the inner surface of the retina that is associated with loss of central vision and decreased visual acuity [70]. In spite of being one of the main causes for vitreoretinal surgery and having a high prevalence [148], it does not exist a screening procedure for diagnosing epiretinal membranes.

Currently, the gold standard for diagnosing ERM is based on the exploration of the fundus by an ophthalmologist, and the confirmation via the analysis of optical coherence tomography images (OCT) [149, 150]. However, acquiring OCT images is an expensive procedure that is not available for all patients. On the contrary, acquiring fundus images is cheaper, and most medical centres have the resources to acquire them. In collaboration with Hospital Vall D’Hebron, we have focused on building a classification model for diagnosing ERM in retinal fundus images. In the rest of this section, we present our approach to tackle this image classification task.

Materials and methods

The ERM dataset employed in this work was created from retinal images of a private database, a nationwide database that collected retinal information from patients attending to optometrists. Images of the database were acquired using different non-mydratic fundus cameras, all of them approved by the National Health Service for Diabetic Screening in the UK [151]. Optometrists were instructed to perform posterior pole retinal photography, centered on the macula and including the optic disc and vascular arcades [152].

Table 2.10: Architectures and backbones employed in our study for diagnosing

Architecture	Backbones
Resnet	34, 50, 101
Resnest	26, 50, 50.4s2x40, 101
EfficientNet	B0-B3
ViT	ViT-B/16-244, ViT-B/16-R50-384
Deit	ViT-B/16-384
NasNet	050
HRnet	w32, w40, w44, w48, w64

The ERM dataset consists of 4081 images (2108 positive samples, and 1973 negative samples) with a size of 299×299 , and it was randomly split using an 80% of the images for training, and a 20% for testing. Furthermore, a 10% of the training dataset was employed for validation in order to adjust the hyperparameters of the models.

We have conducted a thorough study of several families of Deep Learning architectures for diagnosing ERM. The studied architectures, summarized in Table 2.10, included 3 manually designed convolutional neural networks (namely, ResNet [12], ResNeSt [153] and HRnet [145]), 2 architectures found by neural architecture search (EfficientNet [13] and NasNet [154]); and 2 transformer-based architectures that are ViT [155], and its training efficient version, Deit [156]. All the networks used in our experiments were implemented in PyTorch [37], and have been trained thanks to the functionality of the fastai library [38] using a GPU Nvidia RTX 2080 Ti, and using the cross entropy loss function. In order to train the different models, we considered 4 approaches: baseline models, CycleGAN augmentation, state-of-the-art bag of tricks, and transfer learning from a close domain.

First of all, and in order to establish a baseline for our models, we have used the fine-tuning method presented in [38]. This is a two-stage procedure that starts from a model pretrained in the ImageNet challenge. Moreover, we employed early stopping based on monitoring the validation loss, and data augmentation [39] to prevent overfitting. In addition to the classical data augmentation techniques employed for training our baseline models, we have also studied an approach that consists in using a Generative Adversarial Network (GAN) to synthesize new retinal images [157]. In particular, we trained a CycleGAN model [49] that allowed us to synthesize ERM images from healthy images and viceversa (1652 healthy images, and 1622 ERM images were generated using this procedure). The CycleGAN model was trained using the UPIT library³ for 15 epochs and using the learning rate suggested by the algorithm presented in [158]. The generated images were combined with the original dataset and used for training the models by employing the same procedure presented in the previous paragraph. As we mentioned in the previous section, CycleGAN has already been used to deal with the domain shift problem but for this project it has been used as a data augmentation technique. This can be seen as a first step towards including this technology in CLoDSA.

In the third set of experiments, we employed a bag of “tricks” that have been successfully employed in the literature to improve the performance of deep classification

³<https://github.com/tmabraham/UPIT>

models. First of all, we replaced the Adam optimization algorithm, the by-default optimizer used in fastai, with the Ranger algorithm, which combines ideas from the RAdam optimization algorithm [159] and the Lookahead optimizer [160]. Moreover, we used two regularization techniques that are Label Smoothing [82] and MixUp [161]. Finally, we applied the cyclical learning rate policy for convergence proposed in [162]. In order to identify the benefits provided by each trick, an ablation study was conducted.

The last approach that we explored to train our models was based on the fact that transfer learning produces better results when there is a close relation between the source and target task. Hence, we started by training the models with the RIADD dataset [163] (a dataset of 8289 images for multi-disease detection on retinal images); and, subsequently, we fine-tuned the models for our ERM dataset. It is worth mentioning that the models trained for the RIADD dataset did not aim to detect the multiple diseases, but we simplified the problem to determine whether the retinal images were healthy. The RIADD’s models were trained using the procedure presented for the baseline approach.

Finally, and in order to further improve the performance of our models, we employed ensemble methods. Namely, we tested the ensemble of several models [164], the application of test-time augmentation [80] (that is, given an image, we created random modifications of such an image, performed predictions on them using a model, and, finally, returned the average of those predictions), and the combination of these two techniques. As we will show in the following section, these ensemble techniques considerably improved the performance of individual models.

Comparing the technologies used in this project with those used in FrImCla we see how the models used in this case are different. This is due to the fact that Deep Learning methods advance very quickly. The models that were the state of the art during the development of FrImCla have been overtaken by new architectures. For this reason, it is necessary to include updated models and technologies in FrImCla to continue obtaining accurate models for each problem, but this remains as further work.

Results

The models trained with the different approaches presented throughout the previous section were evaluated on the testing set using the F1-score as metric, see Table 2.11 for a summary of the results. The rest of this section is devoted to discuss the advantages and disadvantages of each training approach.

We start by analyzing the baseline models. As we can notice from the first column of Table 2.11, the F1-score of most models is under 70%. The exceptions are the family of HRNet models, and the two transformer-based architectures. The most plausible explanation for those results is the high-resolution representation learned by those models in the ImageNet dataset, which is better transferred to this particular context of diagnosing ERM. It is specially remarkable the ViT-B/16-R50-384 model that achieved a F1-score of 81.29%.

We focus now on the results achieved when training the models with the dataset augmented with the images generated with the CycleGAN model. As we can notice from Table 2.11, the results highly vary among models ranging from an improvement of 9% in the Efficientnet-B1 model, to a 23% decay in the HRNet-w64 model. In general, in most models, augmenting the dataset with the images generated by the CycleGAN

Table 2.11: F1-score achieved by the studied architectures using the baseline procedure, the CycleGAN dataset, the bag-of-tricks, and transfer learning from a close domain. Moreover, we include the results obtained by applying test-time augmentation (TTA) to the models fine-tuned from a close domain, and the results for the RIADD dataset. In italics the best model for each approach, and in bold face the best overall model without TTA.

Architecture	Baseline	CycleGAN	Tricks	Transfer	TTA	RIADD
Resnet-34	55.22	55.18	72.21	59.09	65.69	75.04
Resnet-50	49.53	58.04	75.23	72.18	72.63	73.93
Resnet-101	53.04	46.53	71.85	72.20	72.20	68.38
Resnest-26	55.18	53.59	73.62	62.68	66.36	75.57
Resnest-50	56.02	56.22	76.72	49.22	55.57	75.76
Resnest50d_4s2x40d	56.12	61.99	78.36	63.38	68.10	73.05
Resnest101	59.03	49.63	76.31	56.92	64.00	76.07
EfficientNet-B0	51.16	60.47	73.83	67.43	65.05	78.87
EfficientNet-B1	48.62	47.26	70.14	66.09	71.16	79.05
EfficientNet-B2	60.20	49.94	71.98	61.82	65.30	79.19
EfficientNet-B3	56.68	50.20	73.41	66.96	65.67	79.45
VIT-B/16-244	69.41	62.80	72.21	73.13	76.25	83.01
ViT-B/16-R50-384	81.29	62.91	67.39	83.86	84.23	87.44
Deit-B/16-384	74.85	72.11	81.52	76.46	76.77	87.01
Nasnet-050	55.55	49.76	71.30	55.23	50.66	52.65
HRNet-w32	73.74	67.15	80.50	79.22	81.17	87.98
HRNet-w40	71.09	52.53	70.76	84.00	85.52	87.30
HRNet-w44	72.33	60.79	71.30	82.61	83.27	87.50
HRNet-w48	70.60	76.12	73.95	82.17	84.59	86.32
HRNet-w64	73.78	50.27	77.88	83.70	84.35	87.59

had a negative impact. This might occur due to the challenge of producing realistic images with the GAN models [165]; therefore, this approach needs further investigation to be successfully applied.

On the contrary to the results achieved with the augmented CycleGAN dataset, a clear benefit is obtained with the bag of tricks. Thanks to the set of applied tricks, all the architectures were able to achieve a performance over 70%, we can notice that there is not a single technique, or combination of techniques, that always produce the best results. However, the usage of Label Smoothing and MixUp as regularization techniques consistently produced good results. It is also worth mentioning that the benefits obtained with each individual technique did not stack when combined with other techniques. This hinders the applicability of this bag of tricks since lots of experiments must be conducted to find which methods should be applied to produce the best result for each architecture.

An approach that served to improve most base models, and did not require so many experiments as the bag of tricks, is the application of transfer learning from the RIADD dataset. Pretraining the models with such a dataset, and then fine-tuning them for the ERM dataset achieved a mean improvement of 7.57%. There were only 3 models which performance decayed using this approach, and some models improved more than a 20%. The architectures that took more advantage of this approach were those from the HRNet family, since all of them reached a performance close or even higher than 80%. In fact, the best overall individual model was obtained with the HRNet-w40 architecture with a

F1-score of 84%.

We also analyzed how the performance of the individual models could be improved thanks to the application of ensemble methods. The ensemble of the 5 best models achieved a F1-score of 84.76%; that is an improvement of 0.76% regarding the best individual model. Since, 4 out of 5 of the best models belonged to the same family, we also tested the ensemble of the best individual model of each family; however, the F1-score obtained by such an ensemble was 81.01%, worse than the best individual model. Moreover, we analyzed the impact of test-time-augmentation. This technique was applied to each individual model built using the close transfer approach, and, as we can notice in Table 2.11, the majority of models improved thanks to it (namely, a mean improvement of 3%, and only the performance of 3 models decayed). The best result was again obtained with the HRNet-w40 model with an improvement of 1.52%. Finally, we combined the ensemble of the output produced by the test-time augmentation of the 5 best models, and this produced a F1-score of 86.82%, the best overall result.

Conclusions

The fifth contribution of this work is an open source application that studied several approaches to build deep learning models for diagnosing epiretinal membrane. The best results, with a F1-score of 86.82%, was achieved by using the HRNet and transformer-based architectures, and combining 3 techniques (transfer learning from the RIADD dataset, test-time augmentation and model ensemble).

The next contribution is another collaboration with a hospital. Again, we will use what we have learned previously to solve another real-life problem that is the analysis of motility images.

2.4.2 Analysis of Bacteria on Motility Images

Historically, infectious diseases have been a major cause of mortality and, nowadays, remain as an important problem not only in human, but also in animal and plant health. The discovery of antibiotics changed medical practice by significantly decreasing the morbidity and mortality associated with bacterial infection. However, the rapid emergence, spread and persistence of antimicrobial-resistant microorganisms is a public health problem all over the globe, and possibly one of the major challenges nowadays. Furthermore, the limited therapeutic alternatives to combat them are aggravating the problem [166, 167]. Among the big arsenal of bacterial virulence factors, bacterial surface motility has been regarded as a pathogenicity element because it is essential for many biological functions, such as the search for nutrients, sexual reproduction, but also for the spreading of diseases. Motility is involved in movement between body compartments, host cell adherence, colonization, formation of biofilms, and bacterial survival and persistence [168–170].

Investigations on the motility of bacteria are crucial to understand chemotaxis, biofilm formation and virulence in general. To identify a motile strain in the laboratory, the bacterial spread is observed on media solidified with agar. Up to now, bacteria spread was either measured with a ruler [171], or by using programs like ImageJ [172]

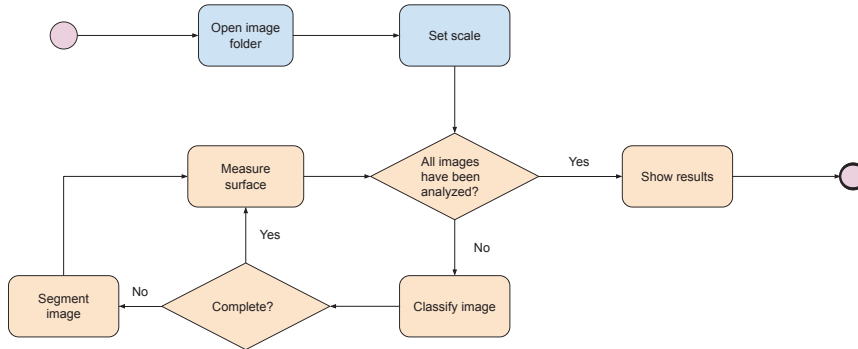


Figure 2.10: Workflow of MotilityJ. Blue components are steps that must be carried out by the user; whereas, orange components are steps automatically conducted by MotilityJ

or Photoshop [173] that allow users to manually draw a region that is subsequently measured. This is a tedious, time-consuming and subjective task that we have automatized by means of Deep Learning techniques. The problem of detecting and measuring the region covered by a bacteria in an image can be framed as a segmentation task. To tackle this task, we have faced several challenges of Deep Learning methods presented throughout this memoir; for instance, the amount of annotated images that is necessary to construct Deep Learning models. In addition, we have deal with a problem related to the deployment of Deep Learning models. Namely, Deep Learning models can be mainly employed inside the framework where they were built; however, non-expert users might find challenging to use those frameworks. In collaboration with the Biomedical Research Center of La Rioja (CIBIR), we have address those limitations by building a standalone application called MotilityJ that automatically classifies and segments motility images by using Deep Learning models. In the rest of this section, we provide an overview of the key components of MotilityJ, and how we have addressed the main challenges of the development of this tool.

MotilityJ

The workflow to employ MotilityJ, diagrammatically described on Figure 2.10, consists of the following steps. First of all, the user selects a folder containing the images to analyze. Then, in order to analyze the images, the user must provide the scale of the images (in terms of the size of the Petri dish). After that, the images are analyzed by first classifying them; and, subsequently, segmenting those that are classified as incomplete. Finally, the analyzed images are shown to the users, where they can edit the annotation produced for each image, and see the area of the bacterial spread. For each image, the results are summarized in the format of a table (in the bottom part of the interface) and can be exported to an Excel file.

We have developed an image processing pipeline that automatically annotates the motility images. Subsequently, such annotations were validated and corrected by experts.

It is worth noting that experts were still needed to be sure about the validity of the annotations; however, they did not need to manually annotate each image; then, the burden of creating an annotated dataset of images was considerably reduced. Finally, using the annotated dataset, we have built a Deep Learning pipeline for segmenting motility images.

The image processing pipeline for annotating motility images was split into two components: a classification procedure, and a segmentation procedure. We explain how these two steps were designed to semi-automatically annotate the 2772 motility images of our dataset. The pipeline has been implemented as an ImageJ's plugin that is available at [71].

The classification procedure was in charge of distinguishing images where the bacteria has covered completely the Petri dish (from now on, these images are called complete images) from those images where the bacteria only covers part of the dish (from now on, incomplete images). This distinction is relevant since complete images do not require any further processing, and the size of the covered area can be directly obtained. We based the procedure to classify the images on the fact that, for incomplete images, the mean intensity of the pixels inside the Petri dish that are close to the border is similar to the mean intensity of the region outside the Petri dish; whereas, for complete images, the mean intensity of the pixels inside the Petri dish that are close to the border is lower than outside the Petri dish. Hence, for this task, we fixed an annulus close to the border of the Petri dish, determined by the Hough transform [174], and compared it with the intensity outside the dish. Using this classification procedure, a 95.63% of the images were correctly classified, and the experts only had to reorganize, approximately, 100 images. As a result, we produced a dataset that consists of 1305 images from the complete class, and 1467 from the incomplete class. The images from the incomplete class were further processed to produce their associated annotation.

The segmentation procedure for the incomplete images aims to produce a mask for the region that contains the bacteria. Our segmentation algorithm was based on the sequential application of several image processing techniques, such as edge detection or thresholding, and morphological operations such as dilation or erosion. Namely, the procedure can be split into two steps: contour generation and contour refinement.

In order to construct and evaluate the classification model for distinguishing complete and incomplete motility images, the classification dataset was split into a training set and a test set using, respectively, 1817 (80%) and 455 (20%) images of the original dataset; additionally, 181 (10%) images of the training set were used for validation in order to avoid overfitting. After that, we increased the size of the training set by applying several data augmentation techniques. From the training dataset, we fine-tuned several convolutional neural networks pretrained on the ImageNet dataset; namely, the last layer of the convolutional networks was replaced with a sequence of linear layers where batch normalization, dropout and a ReLU activation function were applied. In our experiments, we have trained two ResNet architectures (Resnet 50 and 101), an EfficientNet architecture (EfficientNet B3), and a FBNet architecture [175] using the fastai library and a GPU Nvidia RTX 2080 Ti.

For each architecture, we constructed three models by using two different input image sizes (224×224 and 512×512), and using the progressive resizing approach [38]. In order to set the learning rate for the architectures, we employed the two-stage

Table 2.12: Performance (95% CI) for the test set obtained by each classification model. The best result is highlighted in bold face.

Size	Architecture	Precision	Recall	F1-score
224×224	ResNet-50	100 (100-100)	96.14 (94.55–97.73)	98.03 (96.88–99.18)
	ResNet-101	100 (100-100)	97.62 (96.36–98.88)	98.79 (97.89–99.69)
	EfficientNet-B3	99.38 (98.73–100)	95.84 (94.19–97.49)	97.58 (96.31–98.85)
	FBNet	100 (100-100)	97.62 (96.36–98.88)	98.79 (97.89–99.69)
512×512	ResNet-50	99.39 (98.75-100)	97.03 (95.63–98.43)	98.19 (97.09–99.2*)
	ResNet-101	99.39 (98.75–100)	97.32 (95.99–98.65)	98.35 (97.30–99.40)
	EfficientNet-B3	99.38 (98.73–100)	95.54 (93.84–97.24)	97.42 (96.11–98.73)
	FBNet	98.80 (97.90–99.70)	98.51 (97.51–99.51)	98.66 (97.71–99.61)
Resizing	ResNet-50	100 (100-100)	99.70 (99.25–100)	99.85 (99.53–100)
	ResNet-101	100 (100-100)	97.92 (96.74–99.10)	98.95 (98.11–99.79)
	EfficientNet-B3	97.94 (96.77–99.11)	98.81 (97.92–99.70)	98.37 (97.33–99.41)
	FBNet	100 (100-100)	98.51 (97.51–99.51)	99.25 (98.54–99.96)

procedure presented in [38] and used previously for ERM diagnosis. Finally, we applied early stopping in all the architectures to avoid overfitting.

In order to construct the deep segmentation models from the segmentation dataset, we employed 1154 (76%) images for training, and the remaining 313 (24%), for testing; additionally, 115 (10%) images of the training set were used for validation. As for the classification dataset, we increased the size of the training set by applying several data augmentation techniques. From the training dataset, we fine-tuned several deep-learning segmentation algorithms. Namely, we have trained 5 architectures: U-Net [132] (with a Resnet 34 backbone), DeepLabV3+ [144] (with a Resnet 50 backbone), Mask RCNN [117] (with a Resnet 50 backbone), HRNet-Seg [145] (with an HRNet W30 backbone) and U²-net [132] (with its underlying backbone). All the architectures were trained using the same procedure presented in Section 2.3. The experience acquired building those models will serve as a basis for developing a tool similar to FrImCla and UFOD but oriented towards semantic segmentation.

Results

We analyze the results obtained by the deep models built to classify and segment the motility images. We start by analyzing the results achieved by the constructed classification models in the test set, see Table 2.12. As evaluation metrics we employed precision, recall and F1-score. Using those metrics the model that provides a better trade-off between precision and recall is the ResNet-50 model trained using the resizing approach with a F1-score of 99.85%. We can also notice from the results presented in Table 2.12 the benefits of using resizing since all the models are improved using this approach.

We focus now on the results achieved by the deep segmentation models in the test set, see Table 2.13. The metrics employed to measure the accuracy of the different methods are the Dice coefficient and the Jaccard index. As can be seen in Table 2.13, the best model is obtained using the DeepLabV3+ architecture with a Dice coefficient

Table 2.13: Performance (95% CI) for the test set obtained by each segmentation model. The best result is highlighted in bold face.

Model	Dice coefficient	Jaccard index
DeepLabV3+	95.66 (93.40–97.92)	91.68 (88.62–94.74)
HRNet-seg	95.31 (95.97–97.65)	91.05 (87.89–94.21)
Mask-RCNN	91.18 (88.04–94.32)	83.80 (79.72–87.88)
U-Net	60.14 (54.72–65.56)	43.00 (37.52–48.48)
U ² -Net	66.94 (61.73–72.15)	50.31 (44.77–55.85)

of 95.66% and a Jaccard index of 91.68%. A similar result is also obtained using the HRNet-seg architecture, whereas the other models obtain much worse results.

Conclusions

The final contribution of this work is MotilityJ, an open-source framework that automatically analyzes bacteria spread in motility images. The underlying algorithms of MotilityJ are based on highly accurate Deep Learning models that generate a segmentation comparable to those produced by experts; but, considerably reducing the effort required to obtain them. In addition, MotilityJ provides a simple to use interface for editing the results. Thanks to the development of the Deep Learning models and their deployment in MotilityJ, the analysis of motility images will be faster, less subjective, more reliable and comparable among different laboratories all over the world. The developed tools will help to advance our understanding of the behavior and virulence of bacteria.

As a general summary of all the contributions presenter throughout this chapter, it is worth highlighting the work that has been done in bringing Deep Learning methods closer to the largest possible number of users, whether they are experts or not. To achieve this aim, we have developed AutoML tools to help users without experience in these complex technologies, however these libraries also have a configuration so that advanced users can have a use and configuration adapted to them. Other types of Deep Learning technologies such as GANs have also been studied to solve real problems, as we have shown the ERM images project. Finally, in MotilityJ, Deep Learning models were used for the classification and segmentation of the images.

Chapter 3

Contributions

This chapter provides a description of the research publications where we addressed the problems presented in this memoir. For each paper, we provide its abstract and a table with metadata about it.

3.1 FrImCla: A Framework for Image Classification Using Traditional and Transfer Learning Techniques

Deep Learning techniques are currently the state of the art approach to deal with image classification problems. Nevertheless, non-expert users might find challenging the use of these techniques due to several reasons, including the lack of enough images, the necessity of trying different models and conducting a thorough comparison of the results obtained with them, and the technical difficulties of employing different libraries, tools and special purpose hardware like GPUs. In this work, we present the FrImCla framework, an open-source and free tool that simplifies the construction of robust models for image classification from a dataset of images, and only using the computer CPU. Given a dataset of annotated images, FrImCla automatically constructs a classification model by trying several feature extractors (based both on Transfer Learning and traditional computer vision methods) and Machine Learning algorithms, and selecting the best combination after a thorough statistical analysis. Thus, this tool can be employed by non-expert users to create accurate models from small datasets of images without requiring any special purpose hardware. In addition, in this paper we show that FrImCla can be employed to construct accurate models that are close, or even better, to the state-of-the-art models.

Title	FrImCla: A Framework for Image Classification Using Traditional and Transfer Learning Techniques
Authors	Manuel García-Domínguez, César Domínguez, Jónathan Heras, Eloy Mata and Vico Pascual
Journal	IEEE Access
Impact Factor (2020)	3.367
Rank	Q2
Publisher	IEEE
Volume	8
Issue	-
Pages	53443-53455
Year	2020
Month	March
ISSN	2169-3536
DOI	10.1109/ACCESS.2020.2980798
URL	https://ieeexplore.ieee.org/document/9035496
State	Published
Cites	5
Project webpage	https://github.com/ManuGar/FrImCla
Project stats	72,090 downloads (extracted from https://pepy.tech/)
Author's contribution	The PhD student, Manuel García Domínguez, was the main author of the paper. In this work, the PhD participated in the analysis and design phases of the Frimcla software, and was in charge of implementing, documenting and testing it. In addition, he carried out different experiments with the developed software and analyzed the results obtained. Finally, the doctoral student wrote a large part of the article, revised the parts that he had not written, and was in charge of the entire process of submission and revision to the journal in which the work was published.

3.2 UFOD: An AutoML Framework for the Construction, Comparison, and Combination of Object Detection Models

Object detection models based on Deep Learning techniques have been successfully applied in several contexts; however, non-expert users might find challenging the use of these techniques due to several reasons, including the necessity of trying different algorithms implemented in heterogeneous libraries, the configuration of hyperparameters, the lack of support of many state-of-the-art algorithms for training them on custom datasets, or the variety of metrics employed to evaluate detection algorithms. These challenges have been tackled by the development of UFOD, an automated Machine Learning framework that trains several object detection algorithms (using different underlying frameworks and libraries), compares them, and finally selects the best model or ensembles them. Currently, the most well-known object detection algorithms have been included in our system, and new methods can be easily incorporated thanks to a high-level API. UFOD is available at <https://github.com/ManuGar/UFOD/>

Title	UFOD: An AutoML Framework for the Construction, Comparison, and Combination of Object Detection Models
Authors	Manuel García-Domínguez, César Domínguez, Jónathan Heras, Eloy Mata and Vico Pascual
Journal	Pattern Recognition Letters
Impact Factor (2020)	3.756
Rank	Q2
Publisher	Elsevier
Volume	145
Issue	-
Pages	135-140
Year	2021
Month	May
ISSN	0167-8655
DOI	10.1016/j.patrec.2021.01.022
URL	https://doi.org/10.1016/j.patrec.2021.01.022
State	Published
Cites	1
Project webpage	https://github.com/ManuGar/UFOD
Project stats	-
Author's contribution	The PhD student, Manuel García Domínguez, was the main author of the paper. In this work, the PhD participated in the analysis and design phases of the UFOD software, and was in charge of implementing, documenting and testing it. In addition, he carried out different experiments with the developed software and analyzed the results obtained. Finally, the doctoral student wrote a large part of the article, revised the parts that he had not written, and was in charge of the entire process of submission and revision to the journal in which the work was published.

3.3 CLoDSA: a tool for augmentation in classification, localization, detection, semantic segmentation and instance segmentation tasks

Background. Deep Learning techniques have been successfully applied to bioimaging problems; however, these methods are highly data demanding. An approach to deal with the lack of data and avoid overfitting is the application of data augmentation, a technique that generates new training samples from the original dataset by applying different kinds of transformations. Several tools exist to apply data augmentation in the context of image classification, but it does not exist a similar tool for the problems of localization, detection, semantic segmentation or instance segmentation that works not only with 2 dimensional images but also with multi-dimensional images (such as stacks or videos).

Results. In this paper, we present a generic strategy that can be applied to automatically augment a dataset of images, or multi-dimensional images, devoted to classification, localization, detection, semantic segmentation or instance segmentation. The augmentation method presented in this paper has been implemented in the open-source package CLoDSA. To prove the benefits of using CLoDSA, we have employed this library to improve the accuracy of models for Malaria parasite classification, stomata detection, and automatic segmentation of neural structures.

Conclusions. CLoDSA is the first, at least up to the best of our knowledge, image augmentation library for object classification, localization, detection, semantic segmentation, and instance segmentation that works not only with 2 dimensional images but also with multi-dimensional images.

Title	CLoDSA: a tool for augmentation in classification, localization, detection, semantic segmentation and instance segmentation tasks
Authors	Ángela Casado-García, César Domínguez, Manuel García-Domínguez, Jónathan Heras, Adrián Inés, Eloy Mata and Vico Pascual
Journal	BMC Bioinformatics
Impact Factor (2020)	3.169
Rank	Q2
Publisher	Springer
Volume	20
Issue	-
Pages	-
Year	2019
Month	June
ISSN	1471-2105
DOI	10.1186/s12859-019-2931-1
URL	https://doi.org/10.1186/s12859-019-2931-1
State	Published
Cites	34
Project webpage	https://github.com/joheras/CLoDSA
Project stats	103,855 downloads (extracted from https://pepy.tech/)
Author's contribution	The PhD participated in the development of the CLoDSA library. Specifically, he participated in the analysis and design phases of this library. In addition, he implemented, documented, and tested the data augmentation module for object detection models. Finally, the doctoral student wrote part of the article (the one referring to the development that he had carried out), revised the rest of the article, and was involved in the revision process carried out by the journal in which the article was finally published.

3.4 Neural Style Transfer and Unpaired Image-to-Image Translation to deal with the Domain Shift Problem on Spheroid Segmentation

Background and objectives. Domain shift is a generalisation problem of Machine Learning models that occurs when the data distribution of the training set is different to the data distribution encountered by the model when it is deployed. This is common in the context of biomedical image segmentation due to the variance of experimental conditions, equipment, and capturing settings. In this work, we address this challenge by studying both neural style transfer algorithms and unpaired image-to-image translation methods in the context of the segmentation of tumour spheroids.

Methods. We have illustrated the domain shift problem in the context of spheroid segmentation with 4 Deep Learning segmentation models that achieved an IoU over 97% when tested with images following the training distribution, but whose performance decreased up to an 84% when applied to images captured under different conditions. In order to deal with this problem, we have explored 3 style transfer algorithms (NST, deep image analogy, and STROTSS), and 6 unpaired image-to-image translations algorithms (CycleGAN, DualGAN, ForkGAN, GANILLA, CUT, and FastCUT). These algorithms have been integrated into a high-level API that facilitates their application to other contexts where the domain-shift problem occurs.

Results. We have considerably improved the performance of the 4 segmentation models when applied to images captured under different conditions by using both style transfer and image-to-image translation algorithms. In particular, there are 2 style transfer algorithms (NST and deep image analogy) and 1 unpaired image-to-image translations algorithm (CycleGAN) that improve the IoU of the models in a range from 0.24 to 76.07. Therefore, reaching a similar performance to the one obtained with the models are applied to images following the training distribution.

Title	Neural Style Transfer and Unpaired Image-to-Image Translation to deal with the Domain Shift Problem on Spheroid Segmentation
Authors	Manuel García-Domínguez, César Domínguez, Jónathan Heras, Eloy Mata and Vico Pascual
Journal	Arxiv
Impact Factor (2020)	-
Rank	-
Publisher	arXiv
Volume	abs/2112.09043
Issue	-
Pages	-
Year	2021
Month	12
ISSN	-
DOI	10.48550/ARXIV.2112.09043
URL	https://arxiv.org/abs/2112.09043
State	-
Cites	-
Project webpage	https://github.com/ManuGar/ImageStyleTransfer
Project stats	-
Author's contribution	The PhD student, Manuel García Domínguez, was the main author of the paper. In this work, the PhD participated in the analysis and design phases of the Image Style Transfer software, and was in charge of implementing, documenting and testing it. In addition, he carried out different experiments with the developed software and analyzed the results obtained. Finally, the doctoral student wrote a large part of the article, revised the parts that he had not written, and is in charge of the entire process of submission and revision.

3.5 Prediction of Epiretinal Membrane from Retinal Fundus Images Using Deep Learning

An epiretinal membrane (ERM) is an eye disease that can lead to visual distortion and, in some cases, to loss of vision. Screening retinal fundus images allows ophthalmologists to early detect and diagnose this disease; however, the manual interpretation of images is a time-consuming task. In spite of the existence of several computer vision tools for analysing retinal fundus images, they are mainly focused on the diagnosis of diabetic retinopathy and glaucoma. In this work, we have conducted a thorough study of several Deep Learning architectures, and a variety of techniques to train them, in order to build a model for automatically diagnosing ERM. As a result, we have built several models that can be ensembled to achieve a F1-score of 86.82%. The lessons learned in this work can serve as a basis for the construction of Deep Learning models for diagnosing other eye diseases.

Title	Prediction of Epiretinal Membrane from Retinal Fundus Images Using Deep Learning
Authors	Ángela Casado-García, Manuel García-Domínguez, Jónathan Heras, Adrián Inés, Didac Royo and Miguel Ángel Zapata
Journal	Lecture Notes in Computer Science
Impact Factor (2020)	0.25 (SJCR)
Rank	Q3 (SJCR)
Publisher	Springer International Publishing
Volume	12882
Issue	-
Pages	3-13
Year	2021
Month	September
ISSN	0302-9743
DOI	10.1007/978-3-030-85713-4_1
URL	https://link.springer.com/chapter/10.1007/978-3-030-85713-4_1
State	Published
Cites	-
Project webpage	https://github.com/CoVUR/ERM
Project stats	-
Author's contribution	The PhD participated in the development of the library, available at https://github.com/CoVUR/ERM . Specifically, he participated in the analysis and design phases of this library. In addition, he implemented, documented, and tested the creation of the GANs models for the image generation for data augmentation using CycleGAN. Finally, the doctoral student revised the article, and was involved in the revision process carried out by the journal in which the article was finally published.

3.6 MotilityJ: An open-source tool for the classification and segmentation of bacteria on motility images

Background and objectives. Infectious diseases produced by antimicrobial resistant microorganisms are a major threat to human, and animal health worldwide. This problem is increased by the virulence and spread of these bacteria. Surface motility has been regarded as a pathogenicity element because it is essential for many biological functions, but also for disease spreading; hence, investigations on the motility behaviour of bacteria are crucial to understand chemotaxis, biofilm formation and virulence in general. To identify a motile strain in the laboratory, the bacterial spread area is observed on media solidified with agar. Up to now, the task of measuring bacteria spread was a manual, and, therefore, tedious and time-consuming task. The aim of this work is the development of a set of tools for bacteria segmentation in motility images.

Methods. In this work, we address the problem of measuring bacteria spread on motility images by creating an automatic pipeline based on Deep Learning models. Such a pipeline consists of a classification model to determine whether the bacteria has spread to cover completely the Petri dish, and a segmentation model to determine the spread of those bacteria that do not fully cover the Petri dishes. In order to annotate enough images to train our Deep Learning models, a semi-automatic annotation procedure is presented.

Results. The classification model of our pipeline achieved a F1-score of 99.85%, and the segmentation model achieved a Dice coefficient of 95.66%. In addition, the segmentation model produces results that are indistinguishable, and in many cases preferred, from those produced manually by experts. Finally, we facilitate the dissemination of our pipeline with the development of MotilityJ, an open-source and user-friendly application for measuring bacteria spread on motility images.

Conclusions. In this work, we have developed an algorithm and trained several models for measuring bacteria spread on motility images. Thanks to this work, the analysis of motility images will be faster and more reliable. The developed tools will help to advance our understanding of the behaviour and virulence of bacteria.

Title	MotilityJ: An open-source tool for the classification and segmentation of bacteria on motility images
Authors	Ángela Casado-García, Gabriela Chichón, César Domínguez, Manuel García-Domínguez, Jónathan Heras, Adrián Inés, María López, Eloy Mata, Vico Pascual and Yolanda Sáenz
Journal	Computers in Biology and Medicine
Impact Factor (2020)	4.589
Rank	Q1
Publisher	Elsevier
Volume	136
Issue	-
Pages	104673
Year	2021
Month	July
ISSN	0010-4825
DOI	10.1016/j.combiomed.2021.104673
URL	https://www.sciencedirect.com/science/article/pii/S0010482521004674
State	Published
Cites	3
Project webpage	https://github.com/joheras/MotilityJ
Project stats	11.096 (extracted from https://pepy.tech/)
Author's contribution	The PhD participated in the development of the MotilityJ library. Specifically, he participated in the creation of the segmentation models of the library. Finally, the doctoral student revised the article, and was involved in the revision process carried out by the journal in which the article was finally published.

Chapter 4

Conclusions and further work

In this chapter, we present the conclusions of our work, and present our future lines of research.

The main goal of our work was to bring Deep Learning methods closer to a wide variety of users by developing a series of applications that simplify the creation and use of Deep Learning methods. To this aim, we have developed two AutoML tools called FrImCla and UFOD, that facilitate the construction of Deep Learning models for image classification and object detection. FrImCla has several advantages with respect to other AutoML tools for image classification. First of all, FrImCla automatizes the whole pipeline to construct classification models from raw images. In addition, it reduces the amount of data and computational resources that are required to train those classification models thanks to the use of transfer learning. Furthermore, the accuracy achieved by the models constructed with FrImCla is superior to the accuracy obtained using other AutoML tools. In fact, FrImCla models can achieve close results to special purpose models by using a general method that can be applied to any dataset. Thanks to FrImCla, users from several contexts can use state-of-the-art techniques and build accurate models from small datasets, and without requiring any special hardware. In the case of UFOD, this is the first AutoML tool for building Deep Learning models for object detection. The main strength of UFOD is its high-level API that provides a common access point to multiple libraries and frameworks for building object detection models. This feature allows UFOD to automatically search the best model for a custom dataset. As future work in this research line, we want to add libraries and algorithms both to FrImCla and UFOD that would help to find even more precise models, we have seen the benefits of those algorithms when dealing with actual biomedical problems. We also want to add optimization methods such as Bayesian optimization to select the best configuration hyperparameter for each classification and object detection algorithm. Finally, we want to deal with other Computer Vision tasks such as semantic segmentation or anomaly detection; hence, new AutoML tools will be developed.

Both FrImCla and UFOD, and in general any Deep Learning framework and algorithm, produce better models when large datasets of images are provided. This issue has been faced by developing CLoDSA, a library that facilitates the construction of large enough datasets to build models independently of the underlying framework.

This library helps users to automatically apply data augmentation techniques to object classification, location, detection, semantic segmentation, and instance segmentation problems for 2D and multi-dimensional images. This library has been designed using various object-oriented patterns and software engineering principles to facilitate its usability and extensibility. The benefits of applying data augmentation with this library have been tested with different data sets and tools. As a future work in this context, it is planned to add more data augmentation methods such as generating images with a stochastic approach, or the use of GANs to increase the number of images for the mentioned problems. In our work, we have already conducted some experiments for data augmentation using GANs in the context of retinal fundus classification, but more research is still needed to provide a general method applicable to a wide variety of scenarios and tasks.

Independently of the technique or framework used for building Deep Learning models, they generally suffer a generalization problem known as domain shift. We have tackled this problem by developing a framework that facilitates the usage of several style transfer and unpaired image-to-image algorithms to deal with the domain shift problem. The results obtained with our framework show that with these techniques it is possible to handle this generalization problem. In addition, we have seen that the style transfer methods obtain similar results to the unpaired image-to-image translation methods but with the advantage that they do not require any training process, and that they can be implemented using only an image from the source dataset. In our experiments, we have noticed that the methods implemented in our framework have a different impact on the performance of the models; hence, different approach should always be tested. As future work, it remains the task of developing an AutoML tool similar to FrImCla and UFOD that facilitates the automatic comparison of methods to deal with the domain shift problem.

All the technologies and methods studied during our work have been the basis to tackle two real biomedical problems: the diagnosis of diseases from retinal fundus images and the analysis of bacteria on motility images. For the former task, we have studied several Deep Learning models and techniques to build an image classification model able to predict a disease called epiretinal membrane. The best results, with a F1-score of 86.82%, were achieved by using transformer-based architectures, and combining 3 techniques (transfer learning from the RIADD dataset, test-time augmentation and model ensemble). As further work, we plan to extend our work to other retinal diseases. In addition, we aim to further explore the generation of images by using GAN models since this approach did not provide the expected results in this context. For the task of analyzing bacteria on motility images, we have built a user-friendly tool called MotilityJ. This tool uses Deep Learning models to obtain comparable results to those produced by experts, but considerably reducing the effort required to obtain them. Also, MotilityJ has a simple to use interface that allows users to edit the results generated by our models. With this application, the analysis of motility images is done in a more objective, reliable and faster way. As future work for this project, it remains the task of incorporating in MotilityJ the techniques that have been developed to deal with the domain shift problem. This will allow users from different laboratories to carry out studies with different species of bacteria, and using different conditions to capture the motility images.

Finally, we aim to apply our methods and tools in different scenarios. This has

led to a new research line focused on human monitoring. This research line has been opened in collaboration with the *Istituto di Sistemi e Tecnologie Industriali Intelligenti per il Manifatturiero Avanzato* (Bari, Italy). In this context, all the experience gained during the PhD studies will serve to create tools that facilitate the construction and use of models for human monitoring.

Chapter 5

Conclusiones y trabajo futuro

En este capítulo, presentamos las conclusiones de nuestro trabajo, y presentamos nuestras futuras líneas de investigación.

El objetivo principal de nuestro trabajo fue acercar los métodos de Deep Learning a una amplia variedad de usuarios mediante el desarrollo de una serie de aplicaciones que simplifican la creación y el uso de métodos de Deep Learning. Para ello, hemos desarrollado dos herramientas de AutoML denominadas FrImCla y UFOD, que facilitan la construcción de modelos de Deep Learning para la clasificación de imágenes y detección de objetos. FrImCla tiene varias ventajas con respecto a otras herramientas de AutoML para la clasificación de imágenes. En primer lugar, FrImCla automatiza todo el proceso para construir modelos de clasificación a partir de imágenes sin procesar. Además, reduce la cantidad de datos y recursos computacionales que se requieren para entrenar esos modelos de clasificación gracias al uso del aprendizaje por transferencia. Además, la precisión alcanzada por los modelos construidos con FrImCla es superior a la precisión obtenida con otras herramientas de AutoML. De hecho, los modelos FrImCla pueden lograr resultados similares a los modelos de propósito especial mediante el uso de un método general que se puede aplicar a cualquier conjunto de datos. Gracias a FrImCla, los usuarios de varios contextos pueden utilizar técnicas de última generación y construir modelos precisos a partir de pequeños conjuntos de datos y sin necesidad de ningún hardware especial. En el caso de UFOD, esta es la primera herramienta de AutoML para construir modelos de aprendizaje profundo para la detección de objetos. La principal fortaleza de UFOD es su API de alto nivel que proporciona un punto de acceso común a múltiples librerías y frameworks para construir modelos de detección de objetos. Esta función permite a UFOD buscar automáticamente el mejor modelo para un conjunto de datos personalizado. Como trabajo futuro en esta línea de investigación, queremos agregar librerías y algoritmos tanto a FrImCla como a UFOD que ayudarían a encontrar modelos aún más precisos. Hemos visto los beneficios de esos algoritmos cuando se trata de problemas biomédicos reales. También queremos agregar métodos de optimización como la optimización bayesiana para seleccionar el mejor hiperparámetro de configuración para cada algoritmo de clasificación y detección de objetos. Por último, queremos ocuparnos de otras tareas de Visión por Computador como la segmentación semántica o la detección de anomalías; por lo tanto, se desarrollarán nuevas herramientas

de AutoML.

Tanto FrImCla como UFOD, y en general cualquier marco y algoritmo de aprendizaje profundo, producen mejores modelos cuando se proporcionan grandes conjuntos de datos de imágenes. Este problema se ha enfrentado al desarrollar CLoDSA, una biblioteca que facilita la construcción de conjuntos de datos lo suficientemente grandes como para construir modelos independientemente del marco subyacente. Esta biblioteca ayuda a los usuarios a aplicar automáticamente técnicas de aumento de datos a problemas de clasificación, localización, detección, segmentación semántica y segmentación de instancias de objetos para imágenes 2D y multidimensionales. Esta librería ha sido diseñada usando varios patrones orientados a objetos y principios de ingeniería de software para facilitar su usabilidad y extensibilidad. Los beneficios de aplicar el aumento de datos con esta librería se han probado con diferentes conjuntos de datos y herramientas. Como trabajo futuro en este contexto, se planea agregar más métodos de aumento de datos, como la generación de imágenes con un enfoque estocástico, o el uso de GAN para aumentar la cantidad de imágenes para los problemas mencionados. En nuestro trabajo, ya hemos realizado algunos experimentos para el aumento de datos utilizando GAN en el contexto de la clasificación del fondo de retina, pero aún se necesita más investigación para proporcionar un método general aplicable a una amplia variedad de escenarios y tareas.

Independientemente de la técnica o framework utilizado para construir modelos de Deep Learning, estos modelos suelen sufrir un problema de generalización conocido como cambio de dominio. Hemos abordado este problema mediante el desarrollo de un framework que facilita el uso de varios algoritmos de traducción desparejada de imágenes. Los resultados obtenidos con nuestro framework muestran que con estas técnicas es posible manejar este problema de generalización. Además, hemos visto que los métodos de transferencia de estilos obtienen resultados similares a los métodos de traducción desparejada de imágenes pero con la ventaja de que no requieren ningún proceso de entrenamiento y que pueden implementarse utilizando solo una imagen del dataset de origen. En nuestros experimentos, hemos notado que los métodos implementados en nuestro framework tienen un impacto diferente en el rendimiento de los modelos; por lo tanto, siempre se debe probar un enfoque diferente. Como trabajo futuro, queda la tarea de desarrollar una herramienta AutoML similar a FrImCla y UFOD que facilite la comparación automática de métodos para tratar el problema de cambio de dominio.

Todas las tecnologías y métodos estudiados durante nuestro trabajo han sido la base para abordar dos problemas biomédicos reales: el diagnóstico de enfermedades a partir de imágenes de fondo de retina y el análisis de bacterias en imágenes de motilidad. Para la primera tarea, hemos estudiado varios modelos y técnicas de Deep Learning para construir un modelo de clasificación de imágenes capaz de predecir una enfermedad llamada membrana epirretiniana. Los mejores resultados, con una puntuación F1 del 86,82%, se lograron mediante el uso de arquitecturas basadas en transformadores y la combinación de 3 técnicas (aprendizaje de transferencia del conjunto de datos RIADD, aumento del tiempo de prueba y conjunto de modelos). Como trabajo adicional, planeamos extender nuestro trabajo a otras enfermedades de la retina. Además, nuestro objetivo es explorar más a fondo la generación de imágenes mediante el uso de modelos GAN, ya que este enfoque no proporcionó los resultados esperados en este

contexto. Para la tarea de analizar bacterias en imágenes de motilidad, hemos creado una herramienta fácil de usar llamada MotilityJ. Esta herramienta utiliza modelos de Deep Learning para obtener resultados comparables a los producidos por expertos, pero reduciendo considerablemente el esfuerzo necesario para obtenerlos. Además, MotilityJ tiene una interfaz fácil de usar que permite a los usuarios editar los resultados generados por nuestros modelos. Con esta aplicación, el análisis de imágenes de motilidad se realiza de una forma más objetiva, fiable y rápida. Como trabajo futuro de este proyecto, queda la tarea de incorporar en MotilityJ las técnicas que se han desarrollado para tratar el problema de cambio de dominio. Esto permitirá a los usuarios de diferentes laboratorios realizar estudios con diferentes especies de bacterias y utilizar diferentes condiciones para capturar las imágenes de motilidad.

Finalmente, nuestro objetivo es aplicar nuestros métodos y herramientas en diferentes escenarios. Esto ha dado lugar a una nueva línea de investigación centrada en la monitorización humana. Esta línea de investigación ha sido abierta en colaboración con el *Istituto di Sistemi e Tecnologie Industriali Intelligenti per il Manifatturiero Avanzato* (Bari, Italia). En este contexto, toda la experiencia adquirida durante los estudios de doctorado servirá para crear herramientas que faciliten la construcción y uso de modelos para el seguimiento humano.

Bibliography

- [1] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. “Deep learning”. In: *Nature* 521 (2015), pp. 436–444. DOI: 10.1038/nature14539.
- [2] Stuart Russell and Peter Norving. *Artificial Intelligence: A Modern Approach*. PEARSON, 1995.
- [3] Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar. *Foundations of Machine learning*. MIT press, 2012.
- [4] Wei Li, Gai-He Wang, and Amir H. Gandomi. “A Survey of Learning-Based Intelligent Optimization Algorithms”. In: *Archives of Computational Methods in Engineering* 28 (2021), pp. 3781–3799. DOI: 10.1007/s11831-021-09562-1.
- [5] Licheng Jiao et al. “A Survey of Deep Learning-Based Object Detection”. In: *IEEE Access* 7 (2019), pp. 128837–128868. DOI: 10.1109/ACCESS.2019.2939201.
- [6] Richard Szeliski. *Computer Vision: Algorithms and Applications*. Springer, 2010.
- [7] Samet Akcay et al. “Transfer learning using convolutional neural networks for object classification within X-ray baggage security imagery”. In: *2016 IEEE International Conference on Image Processing*. 2016, pp. 1057–1061. DOI: 10.1109/ICIP.2016.7532519.
- [8] Andrea Asperti and Claudio Mastronardo. “The Effectiveness of Data Augmentation for Detection of Gastrointestinal Diseases from Endoscopical Images”. In: *11th International Joint Conference on Biomedical Engineering Systems and Technologies - Volume 2*. 2018, pp. 199–205. DOI: 10.5220/0006730901990205.
- [9] Patrice Simard et al. “Tangent prop-a formalism for specifying selected invariances in an adaptive network”. In: *Neural Information Processing Systems*. Vol. 91. 1991, pp. 895–903.
- [10] Anirudh Choudary et al. “Advancing Medical Imaging Informatics by Deep Learning-Based Domain Adaptation”. In: *Year book of medical informatics* 29.1 (2020), pp. 129–138. DOI: 10.1055/s-0040-1702009.

- [11] Ida Arvidsson et al. “Generalization of prostate cancer classification for multiple sites using deep learning”. In: *IEEE 15th International Symposium on Biomedical Imaging*. 2018, pp. 191–194. DOI: 10.1109/ISBI.2018.8363552.
- [12] Kaiming He et al. “Deep Residual Learning for Image Recognition”. In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pp. 770–778. DOI: 10.1109/CVPR.2016.90.
- [13] Mingxing Tan and Quoc V. Le. “EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks”. In: *International Conference on Machine Learning*. Vol. 97. 2019, pp. 6105–6114. DOI: 10.48550/arXiv.1905.11946.
- [14] Joseph Redmon et al. “You Only Look Once: Unified, Real-Time Object Detection”. In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pp. 779–788. DOI: 10.1109/CVPR.2016.91.
- [15] Shaoqing Ren et al. “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence*. Vol. 28. 2015, pp. 91–99. DOI: 10.1109/TPAMI.2016.2577031.
- [16] David H Wolpert, William G Macready, et al. “No free lunch theorems for optimization”. In: *IEEE transactions on evolutionary computation* 1.1 (1997), pp. 67–82. DOI: 10.1109/4235.585893.
- [17] Isabelle Guyon et al. “AutoML Challenge 2015: Design and first results”. In: *Proc. of AutoML 2015@ICML*. 2015.
- [18] Frank Hutter, Lars Kotthoff, and Joaquin Vanschoren. *Automated Machine Learning: Methods, Systems, Challenges*. Springer, 2019. DOI: 10.1007/978-3-030-05318-5.
- [19] Frank Hutter, Holger H. Hoos, and Kevin Leyton-Brown. “Sequential Model-Based Optimization for General Algorithm Configuration”. In: *Learning and Intelligent Optimization*. 2011, pp. 507–523. DOI: 10.1007/978-3-642-25566-3_40.
- [20] Chris Thornton et al. “Auto-WEKA: Combined Selection and Hyperparameter Optimization of Classification Algorithms”. In: *19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2013, pp. 847–855. DOI: 10.1145/2487575.2487629.
- [21] Matthias Feurer et al. “Auto-sklearn: Efficient and Robust Automated Machine Learning”. In: *Advances in Neural Information Processing Systems*. Vol. 28. 2015, pp. 2755–2763. DOI: 10.1007/978-3-030-05318-5_6.
- [22] Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. “Neural Architecture Search: A Survey”. In: *Journal of Machine Learning Research* 20.1 (2019), pp. 1997–2017.
- [23] Barret Zoph et al. “Learning Transferable Architectures for Scalable Image Recognition”. In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 1–14. DOI: 10.1109/CVPR.2018.00907.

- [24] Haifeng Jin, Qingquan Song, and Xia Hu. “Auto-Keras: An Efficient Neural Architecture Search System”. In: *25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM. 2019, pp. 1946–1956. DOI: 10.1145/3292500.3330648.
- [25] Google. *Google Cloud AutoML*. 2018. URL: <https://cloud.google.com/automl/> (visited on 07/15/2022).
- [26] Ekaba Bisong. “Google AutoML: Cloud Vision”. In: *Building Machine Learning and Deep Learning Models on Google Cloud Platform: A Comprehensive Guide for Beginners*. 2019, pp. 581–598. DOI: 10.1007/978-1-4842-4470-8_42.
- [27] Esteban Real et al. “Regularized Evolution for Image Classifier Architecture Search”. In: *AAAI Conference on Artificial Intelligence*. Vol. 33. 2019. DOI: 10.1609/aaai.v33i01.33014780.
- [28] Chen Sun et al. “Revisiting unreasonable effectiveness of data in deep learning era”. In: *IEEE international conference on computer vision*. 2017, pp. 843–852. DOI: 10.1109/ICCV.2017.97.
- [29] Tsung-Yi Lin et al. “Microsoft COCO: Common Objects in Context”. In: *Computer Vision*. 2014, pp. 740–755. DOI: 10.1007/978-3-319-10602-1_48.
- [30] Ali Sharif Razavian et al. “CNN features off-the-shelf: An astounding baseline for recognition”. In: *IEEE Conference on Computer Vision and Pattern Recognition Workshops*. 2014, pp. 512–519. DOI: 10.1109/CVPRW.2014.131.
- [31] Sinno Jialin Pan and Qiang Yang. “A survey on Transfer Learning”. In: *IEEE Transactions on Knowledge and Data Engineering* 22.10 (2010), pp. 1345–1359. DOI: 10.1109/TKDE.2009.191.
- [32] Stergios Christodoulidis et al. “Multisource Transfer Learning With Convolutional Neural Networks for Lung Pattern Analysis”. In: *IEEE Journal of Biomedical and Health Informatics* 21.1 (2017), pp. 76–84. DOI: 10.1109/JBHI.2016.2636929.
- [33] Mohsen Ghafoorian et al. “Transfer Learning for Domain Adaptation in MRI: Application in Brain Lesion Segmentation”. In: *Medical Image Computing and Computer-Assisted Intervention*. Cham: Springer International Publishing, 2017, pp. 516–524. DOI: 10.1007/978-3-319-66179-7_59.
- [34] Afonso Menegola et al. “Knowledge transfer for melanoma screening with deep learning”. In: *2017 IEEE 14th International Symposium on Biomedical Imaging*. 2017, pp. 297–300. DOI: 10.1109/ISBI.2017.7950523.
- [35] Olga Russakovsky et al. “Imagenet large scale visual recognition challenge”. In: *International journal of computer vision* 115.3 (2015), pp. 211–252. DOI: 10.1007/s11263-015-0816-y.
- [36] François Chollet. *Deep learning with Python*. Manning, 2017.
- [37] Adam Paszkeand others. “Automatic differentiation in PyTorch”. In: *31st International Conference on Neural Information Processing Systems*. 2017.

- [38] Jeremy. Howard and Sylvain. Gugger. “Fastai: A Layered API for Deep Learning”. In: *Information* 11 (2020), p. 108. DOI: 10.3390/info11020108.
- [39] Patrice Simard, Dave Steinkraus, and John C. Platt. “Best practices for convolutional neural networks applied to visual document analysis”. In: *12th International Conference on Document Analysis and Recognition*. Vol. 2. 2003, pp. 958–964. DOI: 10.1109/ICDAR.2003.1227801.
- [40] Alexander B. Jung et al. *imgaug*. <https://github.com/aleju/imgaug>. 2020. (Visited on 07/15/2022).
- [41] Marcus D Bloice, Peter M Roth, and Andreas Holzinger. “Biomedical image augmentation using Augmentor”. In: *Bioinformatics* 35.21 (2019), pp. 4522–4524. DOI: 10.1093/bioinformatics/btz259.
- [42] Alexander Buslaev et al. “Albumentations: Fast and Flexible Image Augmentations”. In: *Information* 11.2 (2020). DOI: 10.3390/info11020125.
- [43] Ekin D Cubuk et al. “Autoaugment: Learning augmentation strategies from data”. In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 113–123. DOI: 10.1109/CVPR.2019.00020.
- [44] Ian Goodfellow et al. “Generative Adversarial Networks”. In: *28th International Conference on Neural Information Processing Systems*. 2014, pp. 2672–2680. DOI: 10.1145/3422622.
- [45] Leon Gatys, Alexander Ecker, and Matthias Bethge. “A Neural Algorithm of Artistic Style”. In: *Journal of Vision* 16 (2016), p. 326. DOI: 10.1167/16.12.326.
- [46] Jason Wang and Luis Perez. “The Effectiveness of Data Augmentation in Image Classification using Deep Learning”. In: *CoRR* abs/1712.04621 (2017). DOI: 10.48550/arXiv.1712.04621.
- [47] Tim Salimans et al. “Improved Techniques for Training GANs”. In: *30th International Conference on Neural Information Processing Systems*. Curran Associates Inc., 2016, pp. 2234–2242.
- [48] Phillip Isola et al. “Image-to-Image Translation with Conditional Adversarial Networks”. In: *2017 IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 5967–5976. DOI: 10.1109/CVPR.2017.632.
- [49] Jun-Yan. Zhu et al. “Unpaired Image-to-Image Translation Using Cycle-Consistent Adversarial Networks”. In: *IEEE International Conference on Computer Vision*. 2017, pp. 2242–2251. DOI: 10.1109/ICCV.2017.244.
- [50] Taesung Park et al. “Contrastive Learning for Unpaired Image-to-Image Translation”. In: *European Conference on Computer Vision*. 2020, pp. 319–345. DOI: 10.1007/978-3-030-58545-7_19.
- [51] Zili Yi et al. “DualGAN: Unsupervised Dual Learning for Image-to-Image Translation”. In: *2017 IEEE International Conference on Computer Vision*. IEEE, 2017, pp. 2868–2876. DOI: 10.1109/ICCV.2017.310.

- [52] Ziqiang Zheng et al. “ForkGAN: Seeing into the Rainy Night”. In: *16th European Conference on Computer Vision*. 2020, pp. 1–16. DOI: 10.1007/978-3-030-58580-8_10.
- [53] Samet Hicsonmez et al. “GANILLA: Generative adversarial networks for image to illustration translation”. In: *Image and Vision Computing* 95 (2020), p. 103886. DOI: 10.1016/j.imavis.2020.103886.
- [54] Sung-Wook Park et al. “Data augmentation using generative adversarial networks (CycleGAN) to improve generalizability in CT segmentation tasks”. In: vol. 9. 2019. DOI: 10.1038/s41598-019-52737-x.
- [55] Emanuele Alberti et al. “IDDA: A Large-Scale Multi-Domain Dataset for Autonomous Driving”. In: *IEEE Robotics and Automation Letters* 5.4 (2020), pp. 5526–5533. DOI: 10.1109/LRA.2020.3009075.
- [56] D. Lacalle et al. “SpheroidJ: An Open-Source Set of Tools for Spheroid Segmentation”. In: *Computer Methods and Programs in Biomedicine* 200 (2021), p. 105837. DOI: 10.1016/j.cmpb.2020.105837.
- [57] Jeremy Irvin et al. “CheXpert: A Large Chest Radiograph Dataset with Uncertainty Labels and Expert Comparison”. In: *Thirty-Third AAAI Conference on Artificial Intelligence*. Vol. 33. 2019, pp. 590–597. DOI: 10.1609/aaai.v33i01.3301590.
- [58] Subhankar Roy et al. “TriGAN: image-to-image translation for multi-source domain adaptation”. In: vol. 32. 2021. DOI: 10.1007/s00138-020-01164-4.
- [59] Wenjun Yan et al. “The Domain Shift Problem of Medical Image Segmentation and Vendor-Adaptation by Unet-GAN”. In: *Medical Image Computing and Computer Assisted Intervention*. 2019, pp. 623–631. DOI: 10.1007/978-3-030-32245-8_69.
- [60] Yue Zhang et al. “Task driven generative modeling for unsupervised domain adaptation: Application to X-ray image segmentation”. In: *International Conference on Medical Image Computing and Computer-Assisted Intervention*. 2018, pp. 599–607. DOI: 10.1007/978-3-030-00934-2_67.
- [61] Jinzheng Cai et al. “Towards cross-modal organ translation and segmentation: A cycle- and shape-consistent generative adversarial network”. In: *Medical Image Analysis* 52 (2019), pp. 174–184. DOI: 10.1016/j.media.2018.12.002.
- [62] Julia Adler-Milstein and Ashish K Jha. “Sharing clinical data electronically: a critical challenge for fixing the health care system”. In: *JAMA* 307.16 (2012), pp. 1695–1696. DOI: 10.1001/jama.2012.525.
- [63] Manuel García-Domínguez et al. “FrImCla: A Framework for Image Classification Using Traditional and Transfer Learning Techniques”. In: *IEEE Access* 8 (2020), pp. 53443–53455. DOI: 10.1109/ACCESS.2020.2980798.

- [64] Manuel García-Domínguez et al. “Jupyter Notebooks for Simplifying Transfer Learning”. In: *Computer Aided Systems Theory – EUROCAST 2019*. Cham: Springer International Publishing, 2020, pp. 215–221. DOI: 10.1007/978-3-030-45096-0_27.
- [65] Manuel García et al. “An On-Going Framework for Easily Experimenting with Deep Learning Models for Bioimaging Analysis”. In: *Distributed Computing and Artificial Intelligence, Special Sessions, 15th International Conference*. Cham: Springer International Publishing, 2019, pp. 330–333. DOI: 10.1007/978-3-319-99608-0_39.
- [66] Manuel García-Domínguez et al. “UFOD: An AutoML framework for the construction, comparison, and combination of object detection models”. In: *Pattern Recognition Letters* 145 (2021), pp. 135–140. DOI: <https://doi.org/10.1016/j.patrec.2021.01.022>.
- [67] César Domínguez et al. “DetectionEvaluationJ: A Tool to Evaluate Object Detection Algorithms”. In: *Computer Aided Systems Theory – EUROCAST 2017*. Cham: Springer International Publishing, 2018, pp. 273–280. DOI: 10.1007/978-3-319-74727-9_32.
- [68] Ángela Casado-García et al. “CLoDSA: a tool for augmentation in classification, localization, detection, semantic segmentation and instance segmentation tasks”. In: *BMC Bioinformatics* 20 (2019). DOI: 10.1186/s12859-019-2931-1.
- [69] Manuel García-Domínguez et al. *Neural Style Transfer and Unpaired Image-to-Image Translation to deal with the Domain Shift Problem on Spheroid Segmentation*. 2021. DOI: 10.48550/ARXIV.2112.09043.
- [70] Ángela Casado-García et al. “Prediction of Epiretinal Membrane from Retinal Fundus Images Using Deep Learning”. In: (2021), pp. 3–13. DOI: 10.1007/978-3-030-85713-4_1.
- [71] Ángela Casado-García et al. “MotilityJ: An open-source tool for the classification and segmentation of bacteria on motility images”. In: *Computers in Biology and Medicine* 136 (2021), p. 104673. DOI: 10.1016/j.combiomed.2021.104673.
- [72] Fabian Pedregosa et al. “Scikit-learn: Machine learning in Python”. In: *Journal of machine learning research* 12 (2011), pp. 2825–2830.
- [73] Piotr Szymański and Tomasz Kajdanowicz. “A scikit-based Python environment for performing multi-label classification”. In: *CoRR* abs/1702.01460 (2017). DOI: 10.48550/arXiv.1702.01460.
- [74] Gary Bradski. “The OpenCV library”. In: *Dr. Dobb’s Journal: Software Tools for the Professional Programmer* 25.11 (2000), pp. 120–123.
- [75] Python software foundation. *CPickle library*. 2018. URL: <https://docs.python.org/2/library/pickle.html> (visited on 07/15/2022).
- [76] Thomas Kluyver et al. “Jupyter Notebooks – a publishing format for reproducible computational workflows”. In: *International Conference on Electronic Publishing*. 2016, pp. 87–90. DOI: 10.3233/978-1-61499-649-1-87.

- [77] Richard. S. Hunter. “Photoelectric Color-Difference Meter”. In: *Journal of the Optical Society of America* 38.7 (1948), p. 661. DOI: 10.1364/JOSA.48.000985.
- [78] Robert M. Haralick, Kumarasamy Shanmugam, and Its’Hak Dinstein. “Textural Features for Image Classification”. In: *IEEE Transactions on Systems, Man and Cybernetics* SMC-3.6 (1973), pp. 610–621. DOI: 10.1109/TSMC.1973.4309314.
- [79] N. Dalal and B. Triggs. “Histograms of Oriented Gradients for Human Detection”. In: *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. Vol. 1. IEEE Computer Society, 2005, pp. 886–893. DOI: 10.1109/CVPR.2005.177.
- [80] Karen Simonyan and Andrew Zisserman. “Very Deep Convolutional Networks for Large-Scale Image Recognition”. In: *CoRR* abs/1409.1556 (2014). DOI: 10.48550/arXiv.1409.1556.
- [81] Gao Huang et al. “Densely connected convolutional networks”. In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 2261–2269. DOI: 10.1109/CVPR.2017.243.
- [82] Christian Szegedy et al. “Rethinking the inception architecture for computer vision”. In: *IEEE conference on computer vision and pattern recognition*. 2016, pp. 2818–2826. DOI: 10.1109/CVPR.2016.308.
- [83] Christian Szegedy et al. “Going deeper with convolutions”. In: *IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2015, pp. 1–9. DOI: 10.1109/CVPR.2015.7298594.
- [84] Pierre Sermanet et al. “OverFeat: Integrated Recognition, Localization and Detection using Convolutional Networks”. English (US). In: *International Conference on Learning Representations*. 2014.
- [85] François Chollet. “Xception: Deep Learning with Depthwise Separable Convolutions”. In: *2017 IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 1800–1807. DOI: 10.1109/CVPR.2017.195.
- [86] Corinna Cortes and Vladimir Vapnik. “Support-Vector Networks”. In: *Machine Learning* 20.3 (1995), pp. 273–297. DOI: 10.1007/BF00994018.
- [87] Thomas Cover and Peter Hart. “Nearest Neighbor Pattern Classification”. In: *IEEE Trans. Inf. Theor.* 13.1 (2006), pp. 21–27. DOI: 10.1109/TIT.1967.1053964.
- [88] Christopher. M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, 1995.
- [89] Jerome H. Friedman. “Greedy function approximation: A gradient boosting machine.” In: *The Annals of Statistics* 29.5 (2001), pp. 1189–1232. DOI: 10.1214/aos/1013203451.
- [90] Peter McCullagh and John A. Nelder. *Generalized Linear Models*. Chapman & Hall, 1989.

- [91] Leo Breiman. “Random Forests”. In: *Machine Learning* 45.1 (2001), pp. 5–32. DOI: 10.1023/A:1010933404324.
- [92] Pierre Geurts et al. “Extremely randomized trees”. In: *Machine Learning* 63 (2006), pp. 3–42. DOI: 10.1007/s10994-006-6226-1.
- [93] Min-Ling Zhang et al. “Binary relevance for multi-label learning: an overview”. In: *Frontiers of Computer Science* 12 (2018), pp. 191–202. DOI: 10.1007/s11704-017-7031-7.
- [94] Jesse Read et al. “Classifier chains for multi-label classification”. In: *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer. 2009, pp. 254–269. DOI: 10.1007/978-3-642-04174-7_17.
- [95] Min-Lin Zhang and Zhi-Hua. Zhou. “ML-KNN: A lazy learning approach to multi-label learning”. In: *Pattern recognition* 40.7 (2007), pp. 2038–2048. DOI: 10.1016/j.patcog.2006.12.019.
- [96] Wei-Jie Chen et al. “MLTSVM: a novel twin support vector machine to multi-label learning”. In: *Pattern Recognition* 52 (2016), pp. 61–74. DOI: 10.1016/j.patcog.2015.10.008.
- [97] Carlos Fernandez-Lozano et al. “Visual complexity modelling based on image features fusion of multiple kernels”. In: *PeerJ* 7 (2019), e7075. DOI: 10.7717/peerj.7075.
- [98] Cha Zhang and Yunqian Ma. *Ensemble Machine Learning: Methods and Applications*. Springer Publishing Company, Incorporated, 2012. DOI: 10.1007/978-1-4419-9326-7.
- [99] John Suckling et al. *Mammographic Image Analysis Society (MIAS) database v1.21*. 2018. URL: <https://www.repository.cam.ac.uk/handle/1810/250394> (visited on 07/15/2022).
- [100] Hiba Chougrad, Hamid Zouaki, and Omar Alheyane. “Deep Convolutional Neural Networks for breast cancer screening”. In: *Computer Methods and Programs in Biomedicine* 157 (2018), pp. 19–30. DOI: <https://doi.org/10.1016/j.cmpb.2018.01.011>.
- [101] Lessage Xavier et al. “Assessing Breast Cancer Screening using recent Deep Convolutional Neural Networks”. In: *International Journal of Computer Assisted Radiology and Surgery* (2018).
- [102] Joe Davison. *DEvol - Deep Neural Network Evolution*. 2018. URL: <https://github.com/joeddav/devol> (visited on 07/15/2022).
- [103] Piero Molino, Yaroslav Dudin, and Sai Sumanth Miryala. “Ludwig: a type-based declarative deep learning toolbox”. In: *CoRR* abs/1909.07930 (2019). DOI: 10.48550/arXiv.1909.07930.
- [104] Nikita Orlov et al. “WND-CHARM: Multi-purpose image classification using compound image transforms”. In: *Pattern Recognition Letters* 29.11 (2008), pp. 1684–1693. DOI: 10.1016/j.patrec.2008.04.013.

- [105] Lior Shamir et al. "IICBU 2008: A Proposed Benchmark Suite for Biological Image Analysis". In: *Medical & Biological Engineering & Computing* 46.9 (2008), pp. 943–947. DOI: 10.1007/s11517-008-0380-5.
- [106] Tsung-Yi Lin et al. "Feature Pyramid Networks for Object Detection". In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 936–944. DOI: 10.1109/CVPR.2017.106.
- [107] Tsung-Yi Lin et al. "Focal Loss for Dense Object Detection". In: vol. 42. 2. 2020, pp. 318–327. DOI: 10.1109/TPAMI.2018.2858826.
- [108] Wei Liu et al. "SSD: Single Shot MultiBox Detectors". In: *European Conference on Computer Vision*. Vol. 9905. Springer, 2016, pp. 21–37. DOI: 10.1007/978-3-319-46448-0_2.
- [109] Maurizio Ferrari Dacrema, Paolo Cremonesi, and Dietmar Jannach. "Are we really making much progress? A worrying analysis of recent neural recommendation approaches". In: *ACM Conference on Recommender Systems*. 2019, pp. 101–109. DOI: 10.1145/3298689.3347058.
- [110] Jonathan Huang et al. "Speed/accuracy trade-offs for modern convolutional object detectors". In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 3296–3305. DOI: 10.1109/CVPR.2017.351.
- [111] Jian Guo et al. "GluonCV and GluonNLP: Deep Learning in Computer Vision and Natural Language Processing". In: 21.1 (2020).
- [112] Lucas Goulart Vazquez and Farid Hassainia. URL: <https://airctic.com/0.12.0/> (visited on 07/15/2022).
- [113] Mingxing Tan, Ruoming Pang, and Quoc V Le. "Efficientdet: Scalable and efficient object detection". In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 10781–10790. DOI: 10.1109/CVPR42600.2020.01079.
- [114] Xuannianz. *EfficientDet (Scalable and Efficient Object Detection) implementation in Keras and Tensorflow*. <https://github.com/xuannianz/EfficientDet>. 2019.
- [115] Zhi Tian et al. "FCOS: Fully Convolutional One-Stage Object Detection". In: *2019 IEEE/CVF International Conference on Computer Vision*. 2019, pp. 9626–9635. DOI: 10.1109/ICCV.2019.00972.
- [116] Chenchen Zhu, Yihui He, and Marios Savvides. "Feature selective anchor-free module for single-shot object detection". In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2019, pp. 840–849. DOI: 10.1109/CVPR.2019.00093.
- [117] Kaiming He et al. "Mask r-cnn". In: *IEEE international conference on computer vision*. 2017, pp. 2961–2969. DOI: 10.1109/ICCV.2017.322.
- [118] Waleed Abdulla. *Mask R-CNN for object detection and instance segmentation on Keras and TensorFlow*. https://github.com/matterport/Mask_RCNN. 2017.

- [119] Tianqi Chen et al. “MXNet: A Flexible and Efficient Machine Learning Library for Heterogeneous Distributed Systems”. In: *CoRR* abs/1512.01274 (2015). URL: <http://arxiv.org/abs/1512.01274>.
- [120] Martín Abadi et al. *TensorFlow: Large-scale machine learning on heterogeneous systems*. 2015. URL: <https://www.tensorflow.org/> (visited on 07/15/2022).
- [121] Kai Chen et al. “MMDetection: Open MMLab Detection Toolbox and Benchmark”. In: *CoRR* abs/1906.07155 (2019). URL: <http://arxiv.org/abs/1906.07155>.
- [122] Zhaowei Cai and Nuno Vasconcelos. “Cascade R-CNN: Delving into High Quality Object Detection”. In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 6154–6162. DOI: 10.1109/CVPR.2018.00644.
- [123] Jifeng Dai et al. “R-FCN: Object Detection via Region-Based Fully Convolutional Networks”. In: *30th International Conference on Neural Information Processing Systems*. 2016, pp. 379–387.
- [124] Mark Everingham et al. “The Pascal Visual Object Classes (VOC) Challenge”. In: *International Journal of Computer Vision* 88.2 (2010), pp. 303–338. DOI: 10.1007/s11263-009-0275-4.
- [125] Andy B. Yoo, Morris A. Jette, and Mark Grondona. “SLURM: Simple Linux Utility for Resource Management”. In: *Workshop on Job Scheduling Strategies for Parallel Processing*. Vol. 2862. 2003, pp. 44–60. DOI: doi.org/10.1007/10968987_3.
- [126] Jan Hosang, Rodrigo Benenson, and Bernt Schiele. “Learning Non-maximum Suppression”. In: *2017 IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 6469–6477. DOI: 10.1109/CVPR.2017.685.
- [127] Ishtiyaque Ahmad. “Automatic detection of diabetic retinopathy from fundus images using image processing and artificial neural network”. In: *Department of computer Science and Engineering* (2019).
- [128] Eduardo Valle et al. “Data, Depth, and Design: Learning Reliable Models for Melanoma Screening”. In: *CoRR* abs/1711.00441 (2017). DOI: 10.48550/arXiv.1711.00441.
- [129] Adrian Galdran et al. “Data-Driven Color Augmentation Techniques for Deep Skin Image Analysis”. In: *CoRR* abs/1703.03702 (2017). DOI: 10.48550/arXiv.1703.03702.
- [130] Xiaosong Wang et al. “ChestX-Ray8: Hospital-Scale Chest X-Ray Database and Benchmarks on Weakly-Supervised Classification and Localization of Common Thorax Diseases”. In: *2017 IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 3462–3471. DOI: 10.1109/CVPR.2017.369.
- [131] Ahmed Fakhry, Hanchuan Peng, and Shuiwang Ji. “Deep models for brain EM image segmentation: novel insights and improved performance”. In: *Bioinformatics* 32.15 (2016), pp. 2352–2358. DOI: 10.1093/bioinformatics/btw165.

- [132] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. “U-Net: Convolutional Networks for Biomedical Image Segmentation”. In: *Medical Image Computing and Computer-Assisted Intervention*. Vol. 9351. 2015, pp. 234–241. DOI: 10.1007/978-3-319-24574-4_28.
- [133] Pauli Virtanen and others. “SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python”. In: *Nature Methods* 17 (2020), pp. 261–272. DOI: 10.1038/s41592-019-0686-2.
- [134] Erich Gamma et al. *Design patterns: elements of reusable object-oriented software*. Pearson Deutschland GmbH, 1995.
- [135] Robert C Martin, James Newkirk, and Robert S Koss. *Agile software development: principles, patterns, and practices*. Vol. 2. Prentice Hall Upper Saddle River, NJ, 2003.
- [136] HDF Group et al. “Hierarchical data format version 5”. In: *The HDF Group* (2019). URL: <https://www.hdfgroup.org/solutions/hdf5/> (visited on 07/15/2022).
- [137] Sivaramakrishnan Rajaraman et al. “Pre-trained convolutional neural networks as feature extractors toward improved malaria parasite detection in thin blood smear images”. In: *PeerJ* (2018). DOI: 10.7717/peerj.4568.
- [138] Joseph Lemley, Shabab Bazrafkan, and Peter Corcoran. “Smart Augmentation Learning an Optimal Data Augmentation Strategy”. In: *IEEE Access* 5 (2017), pp. 5858–5869. DOI: 10.1109/ACCESS.2017.2696121.
- [139] Toan Tran et al. “A bayesian data augmentation approach for learning deep models”. In: *Advances in neural information processing systems*. Vol. 30. 2017, pp. 2794–2803.
- [140] Long Liu et al. “Advanced deep learning techniques for image style transfer: A survey”. In: *Signal Processing: Image Communication* 78 (2019), pp. 465–470. DOI: 10.1016/j.image.2019.08.006.
- [141] Jing Liao et al. “Visual Attribute Transfer Through Deep Image Analogy”. In: *ACM Transactions on Graphics* 36.4 (2017), 120:1–120:15. DOI: 10.1145/3072959.3073683.
- [142] Nicholas Kolkin, Jason Salavon, and Greg Shakhnarovich. “Style Transfer by Relaxed Optimal Transport and Self-Similarity”. In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 10043–10052. DOI: 10.1109/CVPR.2019.01029.
- [143] Sritama Nath and Gayathri R. Devi. “Three-dimensional culture systems in cancer research: Focus on tumor spheroid model”. In: *Pharmacology & Therapeutics* 163 (2016), pp. 94–108. DOI: 10.1016/j.pharmthera.2016.03.013.
- [144] Liang-Chien Chen et al. “Encoder-Decoder with Atrous Separable Convolution for Semantic Image Segmentation”. In: *Computer Vision*. Springer International Publishing, 2018, pp. 833–851. DOI: 10.1007/978-3-030-01234-2_49.

- [145] Jingdong Wang et al. “Deep High-Resolution Representation Learning for Visual Recognition”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2020). DOI: 10.1109/tpami.2020.2983686.
- [146] Xuebin Qin et al. “U²-Net: Going deeper with nested U-structure for salient object detection”. In: *Pattern Recognition* 106 (2020), p. 107404. DOI: 10.1016/j.patcog.2020.107404.
- [147] Hamid Rezaatoughi et al. “Generalized Intersection Over Union: A Metric and a Loss for Bounding Box Regression”. In: *Conference on Computer Vision and Pattern Recognition*. 2019, pp. 658–666. DOI: 10.1109/CVPR.2019.00075.
- [148] Miguel Ángel Zapata et al. “Prevalence of Vitreoretinal Interface Abnormalities on Spectral-Domain OCT in Healthy Participants over 45 Years of Age”. In: *Ophthalmology Retina* 1.3 (2017), pp. 249–254. DOI: <https://doi.org/10.1016/j.oret.2016.11.001>.
- [149] Ying-Chih Lo et al. “Epiretinal Membrane Detection at the Ophthalmologist Level using Deep Learning of Optical Coherence Tomography”. In: *Scientific Reports* 10 (2020), p. 8424. DOI: 10.1038/s41598-020-65405-2.
- [150] Suvimol Reintragulchai et al. “Predicting Chance of Success on Epiretinal Membrane Surgery using Deep Learning”. In: *14th International Joint Symposium on Artificial Intelligence and Natural Language Processing*. 2019. DOI: 10.1109/iSAI-NLP48611.2019.9045159.
- [151] Authors not listed. *Diabetic eye screening: guidance on camera approval*. 2020. URL: <https://www.gov.uk/government/publications/diabetic-eye-screening-approved-cameras-and-settings/diabetic-eye-screening-guidance-on-camera-approval> (visited on 07/15/2022).
- [152] Miguel Ángel Zapata et al. “Telemedicine for a General Screening of Retinal Disease Using Nonmydriatic Fundus Cameras in Optometry Centers: Three-Year Results”. In: *Telemedicine and e-Health* 23.1 (2017), pp. 30–36. DOI: 10.1089/tmj.2016.0020.
- [153] Hang Zhang et al. *ResNeSt: Split-Attention Networks*. 2020. DOI: 10.48550/arXiv.2004.08955.
- [154] Barret Zoph et al. “Learning Transferable Architectures for Scalable Image Recognition”. In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2018, pp. 8697–8710. DOI: 10.1109/CVPR.2018.00907.
- [155] Alexey Dosovitskiy et al. “An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale”. In: *International Conference on Learning Representations*. 2021.
- [156] Hugo Touvron et al. *Training data-efficient image transformers & distillation through attention*. 2021. DOI: 10.48550/ARXIV.2012.12877.

- [157] Alireza Tavakkoli et al. “A novel deep learning conditional generative adversarial network for producing angiography images from retinal fundus photographs”. In: *Scientific Reports* 10.21580 (2020). DOI: 10.1038/s41598-020-78696-2.
- [158] Leslie Smith. “Cyclical Learning Rates for Training Neural Networks”. In: *IEEE Winter Conference on Applications of Computer Vision*. 2017, pp. 464–472. DOI: 10.1109/WACV.2017.58.
- [159] Liyuan Liu et al. “On the Variance of the Adaptive Learning Rate and Beyond”. In: *International Conference on Learning Representations*. 2020.
- [160] Michael Zhang et al. “Lookahead Optimizer: k steps forward, 1 step back”. In: *Advances in Neural Information Processing Systems*. Vol. 32. Curran Associates, Inc., 2019.
- [161] Hongyi Zhang et al. “mixup: Beyond Empirical Risk Minimization”. In: *International Conference on Learning Representations*. 2018.
- [162] Ilya Loshchilov and Frank Hutter. “SGDR: Stochastic Gradient Descent with Warm Restarts”. In: *International Conference on Learning Representations*. 2017.
- [163] Samiksha Pachade et al. “Retinal Fundus Multi-Disease Image Dataset (RFMiD): A Dataset for Multi-Disease Detection Research”. In: *Data* 6.2 (2021), p. 14. DOI: 10.21227/s3g7-st65.
- [164] Cha Zhang and Yunqian Ma, eds. *Ensemble Machine Learning: Methods and Applications*. Springer, 2012.
- [165] Martin Arjovsky and Léon Bottou. “Towards principled methods for training generative adversarial networks”. In: *International Conference on Learning Representations*. 2017.
- [166] Rustam I. Aminov. “A brief history of the antibiotic era: lessons learned and challenges for the future”. In: *Frontiers Microbiology* 8.1 (2010), p. 134. DOI: 10.3389/fmicb.2010.00134.
- [167] Bernardo Riberio da Cunha, Luis P. Fonseca, and Cecilia R. C. Calado. “Antibiotic Discovery: Where Have We Come from, Where Do We Go?” In: *Antibiotics (Basel)* 8.2 (2019), p. 45. DOI: 10.3390/antibiotics8020045.
- [168] Barbara I. Kazmierczak, Maren Schniederberend, and Ruchi Jain. “Cross-regulation of Pseudomonas motility systems: the intimate relationship between flagella, pili and virulence”. In: *Current Opinion in Microbiology* 28 (2015), pp. 78–82. DOI: 10.1016/j.mib.2015.07.017.
- [169] Johanna Haiko and Benita Westerlund-Wikström. “The role of the bacterial flagellum in adhesion and virulence”. In: *Biology (Basel)* 25.2(4) (2013), pp. 1242–1267. DOI: 10.3390/biology2041242.
- [170] Rasika M. Harshey. “Bacterial motility on a surface: many ways to a common goal”. In: *Annual Reviews Microbiology* 57 (2003), pp. 249–273. DOI: 10.1146/annurev.micro.57.030502.091014.

- [171] Xavier Mulet et al. “Biological Markers of *Pseudomonas aeruginosa* Epidemic High-Risk Clones”. In: *Antimicrobial agents and chemotherapy* 57.11 (2010), pp. 5527–5535. DOI: 10.1128/AAC.01481-13.
- [172] Evelyn Sun, Sijie Liu, and Robert E W Hancock. “Surfing Motility: a Conserved yet Diverse Adaptation among Motile Bacteria”. In: *Journal of bacteriology* 200.23 (2018), e00394–18. DOI: 10.1128/JB.00394-18.
- [173] Alexander Yang et al. “Influence of Physical Effects on the Swarming Motility of *Pseudomonas aeruginosa*”. In: *Biophysical journal* 112.7 (2017), pp. 1462–1471. DOI: 10.1016/j.bpj.2017.02.019.
- [174] John Illingworth and Josef Kittler. “The Adaptive Hough Transform”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 9.5 (1987), pp. 690–698. DOI: 10.1109/TPAMI.1987.4767964.
- [175] Bichen Wu et al. “FBNet: Hardware-Aware Efficient ConvNet Design via Differentiable Neural Architecture Search”. In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 10726–10734. DOI: 10.1109/CVPR.2019.01099.